THE Color Computer
MAGAZINE

For Users of TRS-80™ Color Computers

The Electronic Businessman

What Radio Shack's Color Computer Can Offer

- **Colorful Business Practices** by William Barden, Jr.
- **Investor's Analyst for Stocks and Bonds**
- **Statistics Made Easy**

Also:
Scripsit Enhancements
and
A Game Called Fly

GROTE ©84

Shack

BREAK

*™ Tandy Corporation

0 71486 02661

page 69

page 37

page 20

page 106

FEATURES

DEPARTMENTS

Cover
"Barden Means Business," by Rich Grote

Business isn't a topic you'd expect to hear connected with our Color Computer — our micro wasn't designed as a business machine, and, let's face it, the kinds of business programs that most businessmen require aren't available for it. But that doesn't mean you *can't* use your Color Computer for your small business, or home business, or the personal business you do in connection with your home and finances. You just have to look harder for good business programs than *would* an IBM PC user, and you have to be a little more ingenious than a PC user knows how to be... but that's what sets us, and our computer, apart, isn't it?

To help in your search for creative business programs, we've gathered some special articles to get you on your way. GOTO 30 to read a few words by Bill Barden on the Color Computer and business; thus fortified, GOTO 37 and 38 for a double-dose of Scripsit improvements. One will speed-up Color Scripsit (for those of you with little time to spare). The second will help you prepare mailing labels. Now let's look at the most serious side of business (or home finance): money. GOTO 63 for a program that will help you analyze market options like a pro, and GOTO 49 to learn how statistics can help you plug this information into a formula that will produce decision-facilitating data.

Once you have your business and finances under control, you'll need a break. This month we have two excellent pastime activities for you: GOTO 89 for Fly!, a Summer Project Winner that will convince you those nasty buzzing critters were created by a CPU; then GOTO 69 for Sorcerer's Puzzles, and see how your logic paths are operating this month.

June brings with it the end of school and Father's Day. Your kids will want to study this month's Color Computing For Kids (GOTO 20) to plan their Father's Day treat, and we should all take a trip with Paul Kimmelman in GOTO School for some summertime possibilities (GOTO 16).

Last month I promised the ends of some series, and here they are: GOTO 98 for the last program for Color Burner, and GOTO 119 for the final half of the Devil's Disk Editor. GOTO 106 for a series that's drawing closer to its end: Jake's ROM dissection, version 1.0.
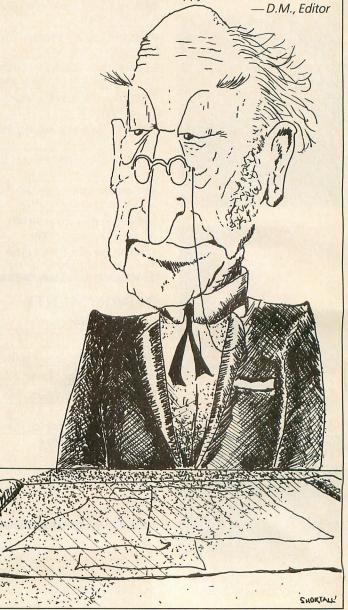
Other projects in the works: GOTO 26 for a neat method to end computer wire confusion; GOTO 96 for a simple explanation of the Boolean functions AND, OR, and NOT. And don't forget our regular features — GOTO 10 (DEFUSR), GOTO 123 (On-Line), GOTO 126 (Reviews), GOTO 140 (New Products) OR you will NOT be on top of the Color Computer world!

A note from the Clear As Mud department: *keyboxes* are those cute little boxes we place on each article to help our readers determine whether their computer set-up can handle a particular program. If you remember to check them, you'll never finish typing a long program only to discover it won't run on your machine beause it requires Disk Basic, which you don't have! Since some of our articles contain several programs with different requirements, the keyboxes will now be found on the first page of each program listing; a quick check of the Table of Contents will tell you memory and language requirements for the article, generally.

You'll also find a note referring to the *Tape Loader*; if you haven't seen the ad (or couldn't decide what it means!), the Tape Loader is a cassette tape produced for us each month, to sell to our readers, containing the major program listings from that month's magazine. Since we are not always able to purchase magnetic media rights to all the programs we publish, or becuase some programs are of such limited interest or odd format that they can't easily by put on tape, not all programs in each month's issue will appear on that month's Tape Loader. The Tape Loader banner on program listings, and the Tape Loader notation preceding the memory requirements on the Table of Contents, will tell you which article's listings will be on this month's Tape Loader.

Now, is that clearer? Have a happy summer!

— *D.M., Editor*

SHORTALL!

INKEY$
The Color Computer Magazine
Highland Mill
Camden, ME 04843

My first original Extended Basic program used all eight graphics pages and more than 16K of RAM. My advice to fellow readers: quit grumbling and upgrade! It's the only way to realize the true potential of your Color Computer.

Ralph Hedenskoog
Gahanna, OH

Managing Data

Just a few lines to thank Keith Baker for the great data management program for the Color Computer in "File It" (January, 1984).

It proved to be every bit as easy to use as indicated, and to modify, with one exception. I have tried several approaches to inserting a line counter for paginating but have been unsuccessful. Perhaps you could supply this modification in a future issue as others may also appreciate this feature for the program.

W. Rau
Elyria, OH

Questions
and Answers

I am writing to ask if I can get listings of the finalist games in your Summer Programming Project and information on the writing of adventure games.

Ron Midthun
Sauk City, WI

No, sorry, we don't own the finalist games. However, we are negotiating to buy many for publication. Keep an eye on future issues. A new series on game programming will be running soon.

— Eds.

Adapting
the Adaptation

Larry Landwehr's article "Remote Override" (March, 1984) is quite concise, but there is a better and easier way than drilling and installing a toggle switch.

In my General Electric cassette recorder, the remote function also shuts off the rewind and fast forward functions. To get around this I desoldered the wires from the remote jack, spliced and soldered them, put heat shrink over the splice and shrunk it. I then traced a wire back from the Play button. I cut and stripped the wire and soldered the wire(s) onto the remote jack.

This fix enables me to fast forward and rewind the tape; but to play or record, it still needs CLOAD/CSAVE or Motor On/ Motor Off commands.

Jim Ayres
Philadelphia, PA

Converting Cashbook

As a rank amateur at writing programs for my 32K Extended Basic Color Computer, I am always pleased to see good useable programs, and most programs I see are in **The Color Computer Magazine.** Your "Cashbook" program (March, 1984), I was distressed to see, is for disk drive systems. Even though I'm not so good at rewriting programs, I undertook the task of converting the CSAVE and CLOAD routines for cassette. I would like to share this with other novices that read the magazine (see Listing 1).

I have also put in a routine to exit the program without having to use the Break key:

```
160 PRINT" <Q>    QUIT PROG-
RAM"
210 CLS:ON INSTR
("AVHC&FISDTQ"),M1$)GOTO10
10,5010,5010,2010,4010,6010,
7010,8010,9010,10010,30000
30000 CLS:END
```

Charles W. Gordon
Ft. Inn, SC

Grabbing Upgrades

When my subscription began a few months ago, I was disappointed that so many of your program listings were not compatible with my family's 16K Color Basic. Radio Shack just slashed the prices on upgrades, and we were among the first to take advantage of the deal, upgrading to 64K Extended Basic.

```
3000 REM *load data from tape*
3010 PRINT"LOADING FILE: ";FS$
3020 OPEN"I",#-1,FS$
3030 INPUT#-1,FS$
3040 INPUT #-1,N$
3050 INPUT#-1,C1
3060 INPUT#-1,C2
3070 DIMSA$(C1),AC$(C1),T(C1)
3080 IF EOF(-1)THEN3170
3090 FOR L=1TOC1
3100 INPUT#-1,B1$(L)
3110 INPUT#-1,B2$(L)
3120 INPUT#-1,B3$(L)
3130 INPUT#-1,B4(L)
3140 INPUT#-1,AC$(L)
3150 INPUT#-1,SA$(L)
3160 NEXTL
3170 GOTO3080
3180 CLOSE#-1:GOTO100
4000 REM *save data to tape*
4010 FT$=FS$
4020 PRINT" YOU ARE IN THE *save* ROUTINE. PREVIOUS DATA
UNDER FILENAME : "FT$" WILL BE LOST": PRINT" AND RE-
PLACED WITH DATA NO W IN MEMORY":PRINT
4030 INPUT" ENTER <S> TO CONTINUE";T$:IFT$<>"S"THEN100
4040 OPEN"0",#-1,FT$
4050 PRINT"DATA FILE NAME IS"
4060 PRINTFT$
4070 PRINT#-1,FT$
4080 PRINT#-1,N$
4090 PRINT#-1,C1
4100 PRINT#-1, C2
4110 FOR L=1TOC1
4120 PRINT#-1,B1$(L)
4130 PRINT#-1,B2$(L)
4140 PRINT#-1,B3$(L)
4150 PRINT#-1,B4(L)
4170 PRINT#-1,AC$(L)
4180 PRINT#-1,SA$(L)
4190 NEXTL
4200 CLOSE#-1
4210 IF RIGHT$(FT$,1)<>"B"THENINPUT" ENTER <B> FOR A
BACKUP                                              COPY
";X$:IFX$="B"THENFT$=LEFT$(FS$,LEN(FS$)-1)+"B":PRINT"BACK-
UP ":GOTO4040
4220 GOTO100
```

Listing 1. Gordon's Cashbook Conversion

INKEY$

Tricky Maneuvers

For the keyboard upgrade explained in "Custom Color" (October, 1983), can't you use a Model III or 4 keyboard instead of a Model 1? If the Model III and 4 keyboards have numeric keypads built onto them then isn't there any way to remove them?

David Ham
Chicago, IL

Yes, you could, if you can cut the circuit board without cracking it while removing the keypad. Model III and 4 keyboards have a hefty metal shield as well; if you can handle a hacksaw, then you could shorten it, but you'd have a problem mounting this end of the keyboard.

— T. Ed.

MC-10 Notes

Thank you, **The Color Computer Magazine** and the authors, Tim McFadden and Doug Kelley of "My MC-10 Speaks ML!," November 1983 issue. I can now save and load machine language programs.

There are three errors in the program listing, however: Line 12 should read GOSUB 1000; Line 110, IF A$< 1 OR A$ > 6 THEN SOUND 1,1:GOTO100 and Line 1040, NEXTC.

For menu selection 2, repeated pressing of W and Z keys could cause an FC error (illegal function call). So, I changed Lines 250 and 260. Line 250 now reads IF A$="W" THEN A=A+512*(A>=512). Line 260 reads IF A$="Z" THEN A=A−512*(A<=65023). Entry of 65024 will display the contents of location 65535 (at the lower right corner) as a period.

Griff Fryer
Sierra Vista, AZ

Happy Hexload

I have only had my Color Computer for 13 months and I'm already tiring of Basic. I don't have an assembler so I had to ignore your great assembly listings until you published "One Finger Hexload," (November, 1983). I wasn't sure what it was at first, but I typed it in anyway. I am sure glad I did! I use it every day and it's great for POKEing in machine code. I typed in the assembly program "Slither" in the November, 1983 issue and was very impressed! The sound was fabulous and so was the speed compared to the Basic version.

I promise to subscribe for as long as **The Color Computer Magazine** is around (forever!).

Jon Alchin
Lompoc, CA

Fix (03,84)

CC Term Fix

In "CC Term" (March, 1984, p. 57), Table 2 omitted the arrow key symbols when describing their function. The table should read (from the top): Up Arrow-$5E, Shift Up Arrow-$5F, Down Arrow-$00, Shift Down Arrow-$5B, Left Arrow-$7F, Shift Left Arrow-$18, Right Arrow-$09, Shift Right Arrow-$5D.

Spazmorg

"Spazmorg" (March, 1984) has a problem. Line 93 was merged with Line 92. This should not be done since the loop that precedes this line has two exits. One falls out on its own, and the other is a voluntary jump made by the user. Line 93 should read: 93 GOTO 16.

Robot Round Up

In the December, 1983 issue of **The Color Computer Magazine** the article entitled 'Topo, Armatron, Poorbot, and "Mim" ' was exactly what I was seeking. I would like to get more information on operating an Armatron by a Color Computer. What can I do?

Stephen Balkum
Arrington, TN

Check the April issue of **The Color Computer Magazine** *for two more articles on Robots.* — *Eds.*

Bulletin Boards

Syracuse BBS

Color-80 of Syracuse is a new Color Computer bulletin board, on-line 24 hours a day, featuring upload, download, news, sports, merchandise, E-Mail and more. All types of computers are welcome. For more information contact: Kris Olmstead, Sysop, Syracuse, NY or call (315)487-0503 (BBS).

Dallas BBS

Careers, a 24-hour BBS, operates out of Dallas, TX using a 64K Color Computer with three disk drives and a 300 baud Hayes Smartmodem. Its theme is career enhancement through education and information. Call (214)692-0513. The board is sponsored by Elkins Institute in Dallas, Inc. and is run privately by John Novocilsky, Jr.

Clubs

CA Club

A Color Computer Club is forming in the Santa Clarita Valley (Newhall / Valencia / Sagus / Canyon Country). For more information, write: SCV Color Computer Club, c/o David Barry, Jr., 23920 Fambrough St., Newhall, CA 91321, or call (805)254-0924.

Erie, PA

Erie TRS-80 User's Group meets monthly, has an extensive software library of public domain programs and publishes a club newsletter. To join call (814)456-4786 or write club president Tom Kuklinski, 320 Maryland Ave., Erie, Pennsylvania 16505

Northern VA

A new Color Computer Club in Northern Virginia meets monthly in the community room of the Manassas Public Library, Manassas, VA. All Color Computer users, beginners to advanced, are welcome. Regular classes in assembly language and Basic are in progress, and several members are informally meeting as an OS-9 SIG. For more information contact Vice President Allan Weinstein, (703) 361-2293 (in the Manassas area) or Logan McMinn (703) 820-0658 (in the greater Washington, D.C. dialing area).

Evansville, IN

For information concerning the Evansville CoCo Club write Brian Broyles, Box 462, Poseyville, IN, 47633 or call (812)874-2210.

DeFUSER

by Terry Kepner

Jake Commander's article, "Dissecting Your ROM," (July, 1983) says it's possible to read Model I tapes into the Color Computer. It sounds interesting and I would like to know how this is done.

—D.K., Macon, GA

I don't have the actual machine code to do the job, but the technique is simple: the Model I tapes store data as discrete pulses placed between timing pulses. Knowing that there are 500 timing pulses coming from the tape every second, all you need do is write a routine to detect the presence of the timing pulse and to check between the pulses for a data pulse. If there's a pulse there, the data bit is a one; if not, it's a zero. Eight pulses together make up one character.

If you want to use this technique to load a Model I program into the Color Computer, you don't have to do the work yourself: Special Projects manufactures "Magic Box," $27.95, a cassette program which does this for you. It's very simple to use: load the program, plug the provided special cable into the right joystick port, plug the other end into the cassette player, put your Model I tape into the player, and start loading.

I've used it many times, and while it sometimes makes mistakes (putting blanks in strange locations), it's an improvement over typing the programs in by hand. Be sure to check ads for any other programs that may do this, as well.

Has anyone developed a football and basketball game for the Color Computer? I mean a real-to-life type game for each sport, with time clock, field goals, point after attempt, and a variety of offensive and defensive plays?

Also, is Radio Shack planning to produce a new 360-degree, heavy-duty version of their joystick? I know Kraft makes one, but I'd like one for my Color Computer.

—S.D., Kansas City, MO

At this point I know of no such programs, and I don't know of anyone planning to do it. That doesn't mean it won't happen.

Yes, by now the new, heavy-duty Kraft joystick should be available in Radio Shack stores — stock number 26-3012, $39.95 each.

Many thanks to R.C. Lake (August, 1983) for a useful Scripsit modification tip. I have two other problems which, I hope, running Scripsit from RAM might solve.

Is it possible, while running Scripsit, to momentarily return to Basic (to change printer font) and go back to Scripsit without losing text?

Can Scripsit be modified to get the program to print French characters (replacing standard keys such as !, @, #, and so forth)? Most of the letters I write are in French, and it makes for a rather unprofessional look to add the accents with a pen after printing.

—M.M., St-Bruno, Canada

That depends on how Basic interacts with the RAM-based Scripsit. If, when you exit Scripsit, Basic doesn't overwrite anything in the text file, you should be able to return to Scripsit with the file intact. The difficulty would be in avoiding the cold-start address to Scripsit, which assumes the text buffer is empty and resets all the pointers to zero. Has anyone solved this problem?

As far as printing French characters is concerned, you have two choices: either write your own keyboard scan routine that intercepts the Scripsit routine before it sends the characters to the text file buffer and replaces the appropriate keys with the characters you want to use, or write a printer patch to translate those standard characters you can do without into the French characters your printer uses. Adding to Lake's modification to accomplish this shouldn't be too hard. Either way, you'll have to remember which characters on the display go with each new French character on the printer.

I'm looking for a reasonably priced Pascal for my 32K non-disk Color Computer. I know of one company with Pascal for the Color Computer, but would like to compare it to others before buying. Surely other companies besides Computerware write Pascal for the Color Computer.

—J.B., Sturgis, MI

Who can I contact about obtaining a version of Pascal for my Color Computer? Try as I might, I can't find anyone who knows where I can look.

—J.S., Norfolk, VA

There are only two companies marketing Pascal directly for the Color Computer (Computerware, 1472 Encinitas Blvd., Encinitas, CA 92024, 714-436-3512 and Deft Systems, Inc., POB 359, Damascus, MD 20872, 800-368-3238), several companies have Pascal for the OS/9 and Flex operating systems.

If you have the OS/9 disk operating system, contact Computerware (address above), Frank Hogg Laboratory (130 Midtown Plaza, Syracuse, NY 13210, 315-474-7856), Microware Systems Inc. (5835 Grand Avenue, Des Moines, IA 50312, 515-279-8844), and OmegaSoft (P.O. Box 70265, Sunnyvale, CA 94086, 408-733-6979) for prices and specific Pascal abilities.

If you have Flex, contact Computerware, Frank Hogg Laboratory, Omega-Soft, and Technical Systems Consultants (1208 Kent Ave., P.O. Box 2570, West Lafayette, IN 47906, 317-463-2502) for prices and specific Pascal abilities.

Unfortunately, all the current versions of Pascal I know of require a disk system, with 64K RAM preferred (usually required).

If I find any other sources, I'll let you know. And be sure to keep an eye on the ad pages for new additions.

After installing eight 64K Motorola chips on my F-board Color Computer, I'm experiencing some RFI while the TV in the other room is clear, even when switched to computer. Do you know of a solution?

—J.B., San Francisco, CA

Circle No. 51 on Reader Service Card

Sounds like the RF shield isn't securely attached to the circuit board. First, visually inspect the metal shield surrounding the SAM and RAM chips to see if it is securely soldered to the circuit board's ground line. The metal cover should fit tightly over the edge of the shield. If it's loose, solder one end of a wire to the cover, the other end to the shield.

Finally, check the RF modulator for good connections to the circuit board. With any loose connections, you could have RFI problems here, especially if the cable to the TV isn't properly grounded to the computer. Similarly, check the modulator box on your TV. I once had severe RFI problems with my computer and couldn't cure the trouble until I replaced the box on the TV (one of the wires to the UHF screws on the TV was worn through and not making good contact).

Do you know any good screen dump programs for the DMP-120 printer?
— **L.W., Syracuse, IN**

The only one I know of is "Screen Print Routine," $19.95, sold by Tom Mix Software (3424 College N.E., Grand Rapids, MI 49505, 616-364-4791). It will send whatever is on your video to an Epson, Microline, or Radio Shack DMP printer. Positive and reverse formats are available, as are horizontal, vertical, small, large, left, right, and center of page layouts. As always, keep an eye on ads for other sources.

My Color Computer won't save any programs to tape and I don't have disk drives. I do know that: the large black plug works; the small grey plug works; the hole that the large grey plug goes into works, because the recorder records with the mike plugged in; and the recorder starts when I type CSAVE. I think either the computer isn't working or the output isn't getting through the cord. Please help!

— **B.H., Franklin, PA**

Before blaming the computer, make sure you're doing everything right. If you aren't using a CCR-81, borrow one. Put the black plug into the cassette EAR jack. Put the small grey plug into the REM jack. Put the large grey plug into the AUX jack.

Now type in a short program and put a blank tape in the recorder. Press the record and play buttons down at the same time. Type CSAVE. When the record- er stops, unplug all plugs and rewind the tape. Now press the play button. You should hear the program as it was stored on the tape (make sure the volume is set at five). If you do hear the program, put the plugs back in, rewind the tape, and try to CLOAD the program. Keep playing with the volume until the program loads.

If you didn't hear anything on the tape, repeat these steps with a different tape recorder; yours could be bad. If neither tape recorders record, the problem could be either your computer or the cassette cords. Radio Shack sells new cords (26-1207, $5.95). Buy a new one, or borrow one from a friend, and repeat the above procedures. If you still can't hear anything on the tape recorders, the problem is definitely with your computer; take it in for repairs.

With a monitor is it possible to have a 64 by 24 screen without wrap-around?

— **C.M., Cut Off, LA**

Yes, but you won't be able to read the characters. The Motorola VDG chip determines video resolution, which is limited to 256 dots across the screen and

192 dots down. Each character is composed of a box of these dots. In standard Radio Shack format of 32 by 16, each box is eight dots wide and 12 dots high. Three dots of width and five dots of height separate characters, leaving five by seven dot characters.

With 64 characters across the screen and 24 lines down, the box is four dots wide and eight dots high. Using one dot to separate them leaves characters three dots wide and seven high. While the height is enough for defining a character, the width is insufficient for legibility.

If you can make do with a screen size of 42 by 24, there's a hardware box that delivers this without eating up any of your computer's RAM. Computer Systems Distributors (P.O. Box 9769, Anaheim, CA 92802, 714-772-1390) sells ColorMate, an expansion box that plugs into the ROMpak port, for $495. It has another ROMpak port built in so you can still use your other ROMpaks without unplugging ColorMate.

I recently purchased a spooler program for my 64K Color Computer (I upgraded it myself). The instructions for a 64K computer say to type EXEC 3589, then says "WARNING! If you have 64K and you are running your Basic in RAM, ... if you type EXEC 3569, your Basic will crash." Does this apply to me?

—P.C., Pasadena, CA

Some people copy the ROM into low RAM, switch the upper 32K bank of RAM in place of the ROM, then copy the ROM from low RAM to high RAM. This gives them the chance to play with and modify Basic and to put data or programs in the highest 16K RAM bank above the ROM, which is usually reserved for ROMpaks.

Since you aren't doing this, the warning doesn't apply to you. For a list of some programs that will let you do this, see the June, 1983 DEFUSR column.

The Radio Shack catalog lists a new Color Computer keyboard. How does it compare with other keyboards mentioned in your magazine? Also, are there any incompatibility problems in connecting a Color 2 disk to an old "grey" Color Computer?

Sears' catalog lists an interface box and connector cable for their Communicator 3 electronic typewriter, which interfaces to Model I and III computers. Can I hook one up to my Color Computer? My wife and daughter want a new typewriter and I'd like a letter quality printer.

—D.B., Albany, OR

Radio Shack's replacement keyboard is the same used on their new Color Computer 2, — not a full-travel standard keyboard, but an improvement over the previous chiclet keyboard. Some people don't like the feel of the new keyboard, preferring standard keyboards advertised in our magazine. The advantage of the Radio Shack keyboard is price. I advise you to go to your local Radio Shack and play with the Color Computer 2 keyboard for awhile and see if you like the way it operates. If you don't, then get one of the standard keyboards.

The new Color 2 disk ROMpak is totally incompatible with the old Color Computer even though the drives may be switched (they're identical except for the exterior cosmetics).

The Sears interface works with parallel port computers and can't be used with the Color Computer unless you buy a serial-to-parallel printer converter box, which increases the price by $70 to $80.

■ ■ ■

GOTO SCHOOL

by Dr. Paul Kimmelman

Rod McCormick
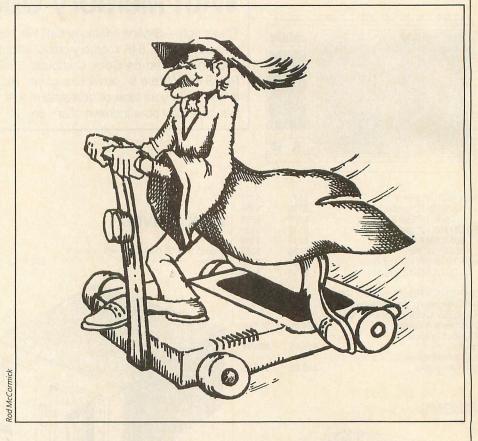
AS SPRING WEATHER BEGINS to arrive and the year-end school bells start to ring, thoughts of summer vacation emerge. Students are happy with no more homework, and parents feel frustrated, wondering how the young ones should be entertained over the summer. One source for summer escape is the increasingly popular "computer camp."

The computer camp comes in all different shapes and sizes. Some are homes away from home, others day-long adventures, and some just a few short hours of respite from the often tedious summer boredom. You can find camps run by entrepreneurs (who may or may not be looking for a "fast buck"), colleges and universities with well-planned programs, computer enthusiasts wanting to share their excitement, and yes, even Radio *Shack with* its well-planned and successful computer workshop for Color Computer users.

Where should you send your budding computer enthusiast? What features should you look for? How much should you spend? These questions must be considered carefully when deciding if computer camp is for your child.

The newspapers and magazines are full of ads claiming their camp will perform wonders for your child. Remember the *basketball* camp that would make your ten-year-old a Larry Bird? Enough for unfounded claims. A computer camp *should* help fulfill your child's needs and interests.

Step one should be a visit to the camp and a talk with the director or instructor. Whether the camp costs $50 or $500, it is going to be responsible for your most precious possession, your child. Sending your child to a poorly run camp will be a horrible experience and will more than likely destroy any interest your child had in computers, not to mention his or her computing potential.

Now that you have visited the camp and see that it meets your standards, what features are most important? Obviously, your budget will establish the parameters: camp costs vary depending on their type. You can send your child to a camp far away from home which will provide not only computer training but those evil moonlight camp stories, as well. If this type of camping experience interests you, then add costs for transportation, food, and lodging.

For those on a limited budget, opportunities should abound in your own town or city. If you are fortunate enough to live near a university or college, rest assured that summer computer opportunities for children exist. I tend to favor the university setting because it offers your child exposure to the world of higher education, and we all tend to hope our children will end up there, right? Usually the college program is run by enthusiastic college students earning their future tuition, along with their professors.

Some local camps have also emerged. Like the college camps they can be well-run and designed to meet the needs of your child. A big caution flag is raised here, however. The word *computer* carries a magical mystique for parents who believe it to be an educational panacea for their child. There are camps that play on the needs of parents who are willing to spend hard-earned cash to provide oppportunities for their children that local schools haven't met. To be certain, visit the camp, talk to the teachers, and ask to see the curriculum — not a bad practice for any of the camps.

Finally, there is our trusted old standby, Radio Shack. Not one to be outdone, the Tandy people have developed their own version of a computer camp.

We'll explore the Radio Shack Computer Camp curriculum later, but for

now let's continue to look at camp features that are very important. Now that we have talked to the director and visited the camp, what's next?

A Camp For Who?

Is the camp really for your child, or are you forcing the experience on him? Don't! How does your child feel about going to camp? Does he show interest in programming activities a la Logo, or does he seem to prefer computer-assisted instructional programs?

Is the length of the camp program suitable? Don't expect your ten-year old to sit for a day with no diversion activities. Some camps offer swimming or recreational activities for the young camper. If the camp lasts for more than three hours, for example, your child will need a pause. Some of the recreational activities may include computer games. Don't be afraid of games — a good blend of computer experiences can make the day more fun and keep your child enthused. Who said learning shouldn't be fun, anyway?

Bear in mind that recreational activities *should* be used for camps of longer duration. A camp that lasts for only a couple of hours daily should get straight to the lesson and remain "on task" full time, but for all-day camps, swimming or volleyball is a pleasant relief.

Now look at staffing. One qualified teacher will be able to work well with a small group, but not with a group of, say, 50. While there is no perfect student-teacher ratio, I suggest a range of one staff member to every 10 or 15 students. Excited children don't like to wait very long for answers to their questions, or for teacher to see their newly created projects.

In many instances there will be a teacher who has assistants to help the students. No problem, but be sure there are enough qualified supervisors for the number of students enrolled. Be sure to ask how many children can be enrolled? Don't stand for "padding" the enrollment unless proper arrangements are made, like more computers, more staff, and more room.

A computer camp should, obviously, have enough computers. Everyone has their own opinion concerning how many computers there should be per student. I suggest no more than two children per computer. In fact, I *prefer* two per computer because it allows children to share their thoughts as well as their excitement. Doesn't one student per computer seem somewhat lonely?

I also believe that every child should have an opportunity to spend some individual time at the computer. Every student needs some individual time to review the lessons and do some homework. This type of time can be scheduled before camp begins, during breaks, and after camp ends each day.

When looking at the curriculum, check to see if it is common to your child's needs. A Logo camp has everyone working on Logo, for example. I would be leery of a camp that enrolls 25 students and teaches Logo, Basic and Pascal. Those languages span a variety of ability levels, and too much is being done for one group. Look for camps that meet a common need.

Chances are you use the Color Computer, so try to find a camp that has at the least *some* Color Computers. It isn't essential to have Color Computers, but children find security in a familiar environment. Putting your little Color Computer user on an IBM may be a little frightening. Don't be afraid to ask if Color Computers are available. Some camps use a variety of brands.

Finally, what curriculum material will be available at the conclusion of the camps? Is there a textbook? Will instruction sheets be given to the camper during camp? After camp you will want your child to be able to apply the material that was learned. Does the instructor provide suggestions for future use of the camp curriculum? Will a future camp be available for those campers who want advanced lessons? Camp won't be useful if all it does is babysit for your child.

Now that we have looked at the different types of camps and explored some important features, let's find out what a Radio Shack Camp will teach your youngster.

Radio Shack Camp

The Radio Shack computer camp program was developed by the Radio Shack Computer Training Programs department. What is the most significant advantage of the Radio Shack camp? Your child will be working on the Color Computer, taught by Color Computer-trained instructors, and working with a well-written Color Computer curriculum. Radio Shack Computer Centers offer their computer camp programs for children at convenient times, during school breaks and summer, for example.

Each instructor works from a comprehensive manual which is accompanied by overhead projections. Using overhead

projections is particularly important, because it includes visual presentations along with the lecture. Not every student learns best by hearing the lesson, so being able to see what is going on is good teaching strategy. Students also spend plenty of time working on the computer, doing hands-on activities — so the blending of activities is very good. What the manual can't do is teach, so once again be sure to meet the teacher before enrolling.

Each camper is given a manual which he can keep when camp is over: no more excuses for not showing mom and dad how to play music or change colors with the Color Computer!

There are two different camps offered by Radio Shack for their Color Computer enthusiast. Both camps were designed for beginners, and particularly for students from 12 – 15 years of age. One camp is specifically for Logo instruction; the other for beginning Color Computer operators. The operator's class also includes a unit on Logo. Each camp lasts two hours per day for five days.

The philosophy of the Color Computer Operators Camp is to provide students who have little or no previous computer experience with a background of computer history, operational skills, and an introduction to Basic and Logo. Your child won't leave the camp an expert, but perhaps he will be enthusiastic enough to pursue the hobby. Maybe he will even write programs whose sales will help support you forever!

Enough dreaming. Let's get back to the operator's class. The camp opens with Radio Shack's software package, "Pioneers in Technology." This presentation gives each camper some background on the history of computers (why do we even need or use computers in our society?).

Following the history of computers, Basic programming is taught. The objectives of the lesson are to explain the meaning of the term Basic, sounds, writing programs that ask questions, counting programs, editing, and changing screen colors.

Camp Secrets

While not divulging camp secrets, can you program your Color Computer to make all the possible sounds? One hint: start at 1. Once you reach the highest note can you bring it back to 1? Get creative — now skip a note or two.

Maybe you aren't an executive secretary and your typing skills leave a little to

be desired. What happens when you make a mistake? If you can edit, the problem can be eliminated. Perhaps whole lines need to be eliminated. Campers will learn preliminary editing techniques.

Learning about PCLEAR, PCLS, PMODE, Screen, Color Sets, Line, Circle, and Paint are all part of the graphics lesson. Be careful: what you learn at camp may not work at home if your computer doesn't have Extended Basic. Some subroutines, string packing, and animation are also offered during the graphics lesson.

Logo, Too

Now that you know how to make a circle, paint a letter, and get some extra memory for those fancy graphics, what's next? You guessed it: Logo. You thought your child was ready to conquer the computer world, and now they start a whole new unit with another computer language.

Logo is an important addition for the Color Computer arsenal. If there was any software that firmly established the Color Computer for educational use, Logo was it. The Logo camp lessons will teach your child about the four modes in Logo, and how to enter and run a Logo procedure. Programming opportunities in Logo are really unlimited, and creativity can produce some exciting programs. If your child's interest is captured, there is a camp devoted to Logo instruction only.

As noted earlier, the Radio Shack camps are for the novice Color Computer user. Computer Center Manager Bill Farkas (Fairlawn, Ohio) offers some final thoughts for parents of campers: "Try to be supportive of your child's interests." Computers play an important role in our society and children are literally growing up with them.

Farkas also suggests that parents look at some of the books offered by Radio Shack, particularly the *Parents Guide To Logo.*" Family computing can be a healthy and happy activity," according to Farkas.

Summer really doesn't have to be a long, boring break from school. In fact, it can be a continuation of some pretty exciting learning opportunities. Computer camps can offer those experiences. They can run for several hours to several weeks and cost from $50 to $500. They can train beginners to experts. Who knows, maybe camp experiences will find your child publishing some very special Color Computer programs.

Happy camping! ■ ■ ■

Color Computing for Kids

INKEY$, a Father's Day gift, and playing your computer like a piano!

---

by Jean Plesser

LAST MONTH WE SURPRISED and delighted some special Moms with electronic greeting cards. This month it's Dad's turn (Father's Day is June 17), and we have a great surprise for him! No, not another greeting card — you can now do that on your own. We're going to challenge Dad with a keyboard match-up game!

Is Dad an "expert," "so-so," or a "beginning" typist? How about you? "Keyboard Match-up" offers three skill levels and everyone can work up to expert with practice. It's a fun game the whole family can enjoy, so after you wrap it up for Dad's day, line up and try your own reflexes and knowledge of the keyboard. Bet your typing will get better!

Before we get to "Keyboard Match-up" we have a few things to learn. This month we're going to play with INKEY$.

INKEY$ (say "Inkey string") is a Color Basic command that tells your computer to check the keyboard and to store as a variable whatever key is pressed. The variable is always a string variable, so the data is always string data. The format is: 10 I$=INKEY$. (The variable's letter can be any that you choose.)

In a program, the INKEY$ command line is read only one time (the computer does not wait for you to respond like it does with an INPUT statement). If you had not pressed a key when the computer read the INKEY$ line, the variable would simply contain no character at all (" ") and the computer would look for more instructions. Use this line to tell the computer that you want something other than nothing.

```
20 IF I$="" GOTO 10
```

Now the computer will keep going back to Line 10 until you press a key. The INKEY$ statement does not print anything on the screen (INPUT prints a question mark), so you must always add a PRINT statement to let the *user* know what to do. The user is whoever is using your program — it may not always be you!

INKEY$ vs INPUT

I've mentioned two differences between the INKEY$ and INPUT statements:

● INKEY$ does not wait for a response from the keyboard before going on to the next statement. INPUT waits for you to press the Enter key.

● INKEY$ does not print anything on the screen. A separate PRINT statement must be used. INPUT prints a question mark on the screen, and a PRINT message can be included in the statement.

Let's use two short programs to demonstrate these differences, first as an INPUT program, then an INKEY$ program.

```
10 CLS
20 PRINT65,"PRESS A KEY THEN PRESS
<ENTER>"
30 INPUT K$
40 PRINT:PRINT"YOU PRESSED- - >";K$
50 GOTO 20
```

You must press the Break key to stop this program and the one that follows; they each enter an endless loop. When you have seen the above program run long enough to remember it, type NEW and enter this program:

```
10 CLS
20 PRINT@65,"PRESS A KEY"
```

```
30 K$=INKEY$
40 IF K$="" "GOTO 30
50 PRINT:PRINT"YOU PRESSED - - >"K$
60 GOTO 20
```

I think you'll agree that the INKEY$ program was faster and more efficient, and it may have seemed like more fun because the computer response was immediate.

To INKEY$ Or
Not To INKEY$

There are two other important differences between INKEY$ and INPUT that will help you determine which statement to use:

● An INKEY$ variable will hold only one character at a time, while an INPUT variable will hold up to 255 characters. When you ran the test programs you became aware of this: the INKEY$ program responded as soon as you pressed a key, but the INPUT program waited until you pressed Enter. If you use INKEY$ and want to collect keyboard responses, they can be added together in one variable (TK$ = TK$ + K$), but we will not be doing that in this lesson.

● The other difference is that INKEY$ responses are stored only as string data. This means that numbers would not have a numeric value. But there is a way in Basic to find the value of a number stored as a string. The function is: V = VAL(K$). This statement tells the computer that V is to equal the VALue of K$. The next program will put the Value function to work.

INKEY-A-Tune

Program Listing 1 will turn the numbered keys on your computer's keyboard into a piano keyboard. The results may be quite different from a baby grand piano, but remember you're not playing a baby grand. You're playing a computer, and as computers go, ours sounds pretty terrific!

I think you'll have fun with "Inkey-A-Tune" as well as learn a few programming tricks. Here's a line-by-line explanation:

Line 10 is a Remark statement that names my program.
Line 20 clears the screen and prints the title at screen location 43.
Lines 30 – 50 print the numbers to press and the notes they will play. This is a user message that helps the user understand the program.
Line 60 prints a message that tells the user that the notes played will be printed on the screen.

Line 70 will print a space at screen location 352, and the semicolon tells the computer to stay on the same line to print whatever follows. This is a way of printing all the notes that are played right across the screen, starting at location 353. The printing is done later in a loop.

Line 80 assigns whatever key is pressed (INKEY$) to the variable M$. A second statement on the same line keeps the computer here until a key has been pressed and M$ equals a character.

Line 90 is an example of the Value function we talked about. We are using the numbers on the keyboard for a purpose, and their number value is important to the program. For example, if M$ = "5" the 5 is only a character to the computer — it has no numeric value: it is string data. The Value command gives the number in a string variable a numeric value, and a numeric variable will then hold the value N = 5.

Line 95 is a test statement using the operators < (less than) and > (greater than). Because we are using only eight notes, numbered from one to eight, a keyboard response that is out of this range is an error. The test statement simply sets N to one whenever an error occurs. If this line were omitted from the program, and N were to contain a larger number, we would get an OD ERROR message. This is the computer's way of telling us we ran out of data — we only have eight sets of data to read.

Line 100 begins a For/Next loop that will repeat whatever number of times N equals. This loop will read data until L = N. The data found at that point is then used by the program.

Line 110 sends the computer to the data statements to read one piece of string data (N$) and one piece of numeric data (T). The second statement on this line tells the computer to go back to Line 100 until L = N.

Line 120 prints the contents of N$ on the screen at the location designated. Remember the semicolon (;)? Continued use of the semicolon prints the notes on one continuous line. The next statement on this line is the SOUND command, and the contents of T are being used for the *tone* portion of this command. The number five is used for the *duration* (length) of the sound, or note.

Line 130 resets the data pointer to the beginning of the data statements. The program then returns to Line 80 and begins again. If we didn't use the command RESTORE, we would get another OD ERROR message when the program repeated and ran out of data to read.

"Inkey-A-Tune" will continue running until you press the Break key. You can

play simple tunes, and if you make up one that you like, you can play it again because the notes you have played are printed on the screen for you.

Keyboard Match-Up

Program Listing 2 is a keyboard skill game that will make a great Father's Day gift for the special Dad in your life. "Keyboard Match-up" is fun, requires skill, and teaches letter placement.

As always, you may want to get help typing it in or do your typing in short session, the CLOAD what you've saved when you're ready to type some more, CSAVE that session's work, and so on. work, and so on.

I've added a few new routines to this program so I'll first list the variables, then explain "Keyboard Match-up."

The variables:

I$ = INKEY$ (keyboard response).
S = the skill level time allowed.
R = a random number used to control a loop that reads data.
L$ = the letter attained by the "random number read loop."
TM = the total number of matches, or games.
T = the time allowed in each skill level.
TC = the total number of correct matches made.

Lines 20 through 80 print a *menu* on the screen. A menu is used to tell the user what the program offers and give a selection. In this case the user will select a skill level. The skill level is selected by pressing 1, 2, or 3; all other keys are ignored.

Lines 90 through 100 examine the contents of I$, assign skill values to S and *GOTO* to the appropriate routine.

Lines 110 through 130 simply keep the game from starting until the user is ready to play. The user indicates he's ready by pressing any key.

Lines 140 through 180 print out the game screen.
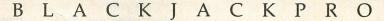
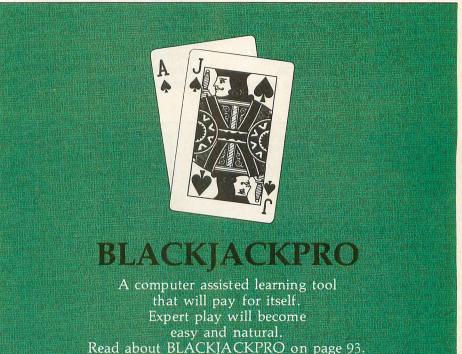Line 200 selects a random number for the variable R, that will be no greater than 26.

Line 210 uses the value of R to read "R" number of data statements. This results in a random letter (L$) that will be used in the game.

Line 230 adds one to the total match counter and makes T equal to the value of S.

Line 240 prints the random letter that the user must match.

Lines 250 through 290 create a sound and a time countdown until a key is pressed or time runs out.

Line 300 prints the contents of K$.

Line 310 examines L$ for a correct match.

Line 330 is an incorrect match routine.

Lines 350 and 360 are for an "out of time" routine.

Line 380 is a correct routine and the total correct counter is added to.

Lines 390 through 460 are executed after any one of the above routines. Totals are printed on the screen and a replay routine using INKEY$ is done.

Lines 470 through 490 are the data — the letters of the alphabet that are randomly selected.

Summer is the time to live it up a little, and the INKEY$ command can liven up your old programs. Also, use these ideas to create new and fun programs on some of your quiet, lazy summer afternoons.

We're going to take a break from programming next month and talk about software — how it's made and why we should protect it from modern-day pirates! See you then and have a happy last day of school!

∎∎∎

Program Listing 1.
A Piano Keyboard

4K Color Basic

```
10 REM--INKEY A TUNE--
20 CLS:PRINT@43,"KEYBOARD FUN"
30 PRINT@98,"THE NUMBERS TO PRES
S FOR EACH NOTE ARE:"
40 PRINT@196,"1=C    2=D    3=E
   4=F"
50 PRINT@260,"5=G    6=A    7=B
   8=C"
60 PRINT@320,"THE NOTES YOU HAVE
   PLAYED ARE:"
70 PRINT@352,;
80 M$=INKEY$:IF M$="" THEN 80
90 N=VAL(M$)
95 IF N<1 THEN N=1 ELSE IF N>8 T
HEN N=1
100 FOR L=1 TO N
110 READ N$,T: NEXT L
120 PRINT N$;: SOUND T,5
130 RESTORE: GOTO 80
140 DATA C,89,D,108,E,125,F,133
150 DATAG,147,A,159,B,170,C,176
```
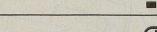
∎∎∎

Program Listing 2.
Keyboard Match-Up

4K Color Basic

```
10 REM---KEYBOARD MATCH-UP---
20 REM--SKILL MENU--
30 CLS:PRINT@37,"*KEYBOARD MATCH
   GAME*"
40 PRINT@106,"SKILL LEVELS:"
50 PRINT@171,"1 = EXPERT"
60 PRINT@235,"2 = SO-SO"
70 PRINT@299,"3 = BEGINNER"
80 PRINT@417,"SELECT A NUMBER--"

90 I$=INKEY$: IF I$="" THEN 80
100 IFI$="1"THEN S=10 ELSE IFI$=
"2" THEN S=15 ELSE IF I$="3" THE
N S=30 ELSE 80
110 CLS:REM--GET READY--
```

```
120 PRINT@229,"PRESS ANY KEY TO
BEGIN"
130 I$=INKEY$: IF I$="" THEN120
140 REM--GAME SCREEN--
150 CLS:PRINT@65,"THE LETTER TO
MATCH BEFORE THE"
160 PRINT@131,"TIME RUNS OUT IS-
-->"
170 PRINT@204,"TIME = "
180 PRINT@262,"YOUR RESPONSE--->
"
190 REM---SET-UP VALUES--
200 R=RND(26)
210 FOR X=1 TO R:READ L$:NEXT X
220 REM--GAME ROUTINE--
230 TM=TM+1: T=S
240 PRINT@152,L$
250 I$=INKEY$
260 SOUND1,1: T=T-1
270 PRINT@211,T
280 IF T=0 THEN 340
290 IFI$="" THEN 250
300 PRINT@280,I$
310 IF I$=L$ THEN 370
320 REM---INCORRECT ROUTINE---
330 SOUND100,5: GOTO 390
340 REM--OUT OF TIME ROUTINE--
350 SOUND 50,10: PRINT@262,"****
OUT OF TIME****"
360 GOTO 390
370 REM---CORRECT ROUTINE---
380 SOUND 200,5: TC=TC+1
390 REM--TOTALS--
400 PRINT@352,"CORRECT MATCHES =
";TC
410 PRINT@384,"TOTAL MATCHES =";
TM
420 REM---REPLAY ROUTINE--
430 PRINT@448,"PLAY AGAIN (Y/N)?
"
440 I$=INKEY$:IF I$="" THEN 440
450 IF I$="Y"THEN 460 ELSE IF I$
="N" THEN END ELSE 440
460 RESTORE: GOTO 150
470 DATA A,B,C,D,E,F,G,H,I,J,K
480 DATA L,M,N,O,P,Q,R,S,T,U,V
490 DATA W,X,Y,Z
```

∎∎∎

DON'T LET YOUR KIDS GO TO SEED THIS SUMMER

With entertaining educational software from TRIAD PICTURES the learning will go on all summer long.

EDUCATION

ALPHABET STEW

Youngsters will LOVE this never-ending "stew" of animated sights and sounds. Even kids as young as 18 months squeal with delight when the computer responds to THEIR touch! Press any key and get an animated picture.
Press the same key again and a DIFFERENT picture appears.
Number keys play musical notes which are displayed on the screen. Kids of all ages will use it for hours!

32K/EXT.   CASS. $22

SUPERFLASH!

Computerized super 'flash cards' -the most effective learning method. A fast, easy way to learn ANYTHING! Enter your own list, up to 300 items! Easily saved on cassette. The computer compliments or corrects as you go. Score and completion time are both displayed so you can play against yourself. Even replay the missed cards. You'll love this one!

16-32K/EXT.   CASS. $14

NUMBER POWER

Help your child gain SPEED AND ACCURACY in addition and subtraction. Students progress at their own rate. Continual computer encouragement and generous review of missed problems.

16K/EXT.   CASS. $14

SPELLING LIST

The ideal way to master spelling. Enter your own words or use your child's spelling list from school. Ample encouragement and review. Save lists on cassette.

16K/EXT.   CASS. $12

DIGIDOTS

Colorful dots and numbers teach addition and subtraction in a unique and entertaining way. A child gains two related subtraction facts for each addition fact learned. With the computer's encouragement students advance at their own rate. Automatic replay of missed problems.

16K/EXT.   CASS. $14

ACCESSORY SOFTWARE
for SPELLING LIST
& SUPERFLASH

Add POWER to your learning with these unique cassettes.

$10 FOR ONE.
$5 FOR EACH ADDITIONAL CASSETTE IF ORDERED AT THE SAME TIME.

PRE—PROGRAMMED WORD PACKAGES

Ready to use with SPELLING LIST (see above). Based on the principle of ASSOCIATION, essential to the learning and retention of facts. Words are grouped according to their letter-patterns.
LEVELS 1 to 4 (specify level) level corresponds approx. to grade

COMMONLY MISSPELLED WORDS

Learn them all . . . easily! Used with SPELLING LIST (see above). A must for the student or adult whose goal is to become more literate.

SUPERFLASH ADDITION

Gain SPEED and ACCURACY with one-and-two-digit numbers. Students progress until they may even be adding two double-digit numbers MENTALLY, and with ease! Must be used with SUPERFLASH (see above).

SUPERFLASH SUBTRACTION

Make your child a whiz at subtracting double-digits . . . in their head! Designed by a Mathematics teacher with an extensive background in educational psychology. Must be used with SUPERFLASH (see above).

SUPERFLASH TIMES TABLES

Learn the "Times-Tables" the EASY, EFFECTIVE way! Covers the tables from 1 X 1, all the way through 12 X 12. Must be used with SUPERFLASH (see above).
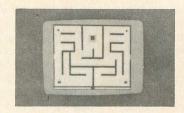
SUPERFLASH DIVISION TABLES

Who ever heard of DIVISION TABLES? So many students do not do well in the higher grades because of this oversight. Must be used with SUPERFLASH (see above).

FRACTIONS = DECIMALS = PERCENT

Teaches 95 different fractional values and their equivalents in DECIMALS, FRACTIONS and PERCENT! Example: $4/5 = .8 = 80\%$. Help your child in this important area. It could even help you! Must be used with SUPERFLASH (see above).

AND WHEN IT'S TIME TO PLAY

SPLAATT!

A delightful new twist on the "combat tank" theme. "SPLAATT" your opponent WITH PAINT before he Splaatt's you! Progress thru seven imaginative "zones" such as 'PEEK—A—BOO', 'HIGH NOON', and 'HELTER SKELTER'. Play against a friend · or the computerized 'SPLAATT—MAN'.

32K/EXT./JOYSTICKS   CASS. $22

TRACER

The fastest action around! This slapbang crash-filled dash for the prize will keep you on the edge of your seats. One or two players.

16K/EXT./JSTKS   CASS. $12

TIME WARP

See how fast you can streak thru the twisting, turning TIME WARP and escape to the worlds beyond. Blast those deadly ALPHA WAVES before they anihilate you!
A VERY colorful and challenging game with STUNNING graphics. Never the same screen twice!

16K/EXT./JSTKS   CASS. $14

JEEPERS REAPERS!

Take a wild ride thru a treasure laden field! Your revved-up reaper gobbles grass as you race for the gold. Play against a human opponent or take on the ominous BLACK REAPER. Beat them to the treasure before the RED PHANTOM snatches it away!
Never the same screen twice! 25 levels of play.

16K/EXT.   CASS. $16

ETCH—A—DOT

Let your imagination run wild! Create your own pictures in a fantasy of color! Even KIDS will have no trouble controlling THIS drawing program. Arrow keys guide the cursor with perfect accuracy. "Blank line" lets you move the cursor without erasing. All 9 colors are at your fingertips · no complicated codes to learn · just hours of FUN. Save your favorite creations on cassette.

16/EXT./JSTKS   CASS. $14

...FROM THE ENTERTAINMENT COMPANY THAT KNOWS EDUCATION

triad pictures corp.

P.O. BOX 1299, SEQUIM, WA 98382

ADD $2 SHIPPING AND HANDLING
(3 OR MORE — WE PAY S/H)

Internal Modem

Reducing wire confusion

by M.K. Marston

A S MY COLOR COMPUTER system grew over the years, I began to realize that I was spending more time being a "boy scout" than I spent programming. After staring at my hopeless tangle of cables, I decided to remove a little clutter. The fact that Radio Shack's videotext machine has a built-in modem, and yet uses the Color Computer case, inspired me to install my modem inside my machine.

After opening my Radio Shack Modem I, I discovered that the circuit board inside is smaller than the case and reasonably flat. A quick measurement of the board revealed that it would fit over the RFI shield inside the computer, with some modifications to the board and the computer case.

If you attempt the following modification you will have to modify the modem and soldering experience is a must. Experience with switches and LEDs is also helpful, but is not absolutely necessary. This modification will also void the warranty on the modem and the computer.

Method

First remove the four screws located on the bottom of the modem. Remove the top cover and you will see the circuit board inside. There are four more screws on the circuit board, attaching it to the bottom half of the case. These must be removed as well. The circuit board should now be free and you may remove it from the case.

To control the modem, the answer/originate and power toggle switch must be removed. If you are sure this will be a permanent modification, you may take a small pair of wire cutters and carefully cut each terminal of the toggle switch.

Try to cut each post as close to the switch case as possible. This becomes a place to solder future wires. If you think you may decide to return the modem to its original state, you will have to remove the switch by carefully heating each terminal, and remove the solder with solder wick or a similar product. Gently break each terminal loose with a small screwdriver and remove the switch. Using a pair of wire cutters, cut each pin of the LEDs on the modem board as close to the base of the LED as possible, leaving as much as possible of the pin on the board (again, for a place to solder the new wires).

You will need two DPDT micro-miniature toggle switches to replace the one you have just removed. Solder a four-teen-inch piece of wire to each terminal on one of the switches. Solder a fourteen-inch piece of wire to all but the top two terminals on the other switch (see Figure 1).
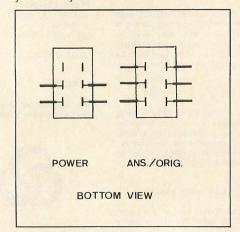
You will also need two LEDs (any of Radio Shack's will work fine). Solder a sixteen-inch piece of wire to each pin on the LED. Be sure to insulate each pin on the LED with electrical tape to prevent shorting.

On the modem circuit board you may either solder four eight-inch pieces of wire to the back of the RS-232 jack, or you may remove the jack, as described with the toggle switch above, and solder the wires directly to the board. If you solder the wires to the back of the jack, add a little solder to each terminal first and then tin each wire. This will ease soldering.

Some modifications to the computer case must now be made. First remove the six screws on the bottom of the computer. The top of the case should be free. Four holes are needed to accommodate the two switches and the two LEDs: the area to the right of the logo, just above the keyboard, is a good place. Drill two holes, four inches from the right side of the case and 5/8 inches up from the black area surrounding the keyboard. The holes should be side by side, about an inch apart. The two DPDT switches will be mounted here. Just above each switch, on the curve in the case, drill two more holes. These are for the two LEDs (see Figure 2). After drilling the holes, mount the switches and LEDs. The switch with the two empty top terminals goes in the left hole.

The plastic post in the middle of the RFI shield must be cut at the shoulder (Figure 3) so the modem circuit board may rest on it. After trimming the post, place a piece of double-sided foam tape on the top of the post to prevent any remaining sharp edges from damaging the underside of the circuit board.

Figure 1. Switch Terminals

POWER   ANS./ORIG.

BOTTOM VIEW
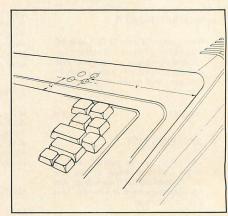
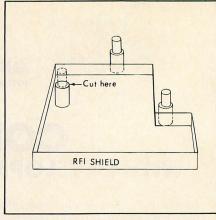*Figure 2. Hole locations for Switches and LEDs.*
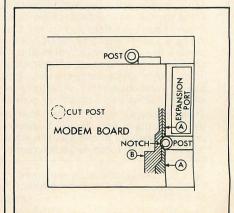
Figure 3. RFI Shield Post

"Test fit" the circuit board by placing it in the computer, over the RFI shield. The back of the circuit board should be even with the back of the plastic frame surrounding the expansion port. There is a small shelf on the frame. This makes a good resting spot for the circuit board, and provides a place to secure it.

On the edge of the board, there is a screw hole. Mark the spot where the hole sits over the shelf. Make a small hole in the shelf at that spot. This will let you use one of the original screws to attach the board at that point. Insert the screw through the circuit board and partially tighten it in the hole on the plastic shelf. The circuit board should be loose enough to move.

Note where the right edge of the circuit board comes in contact with the plastic post at the right side of the RFI shield. Carefully cut a *small* notch in the circuit board at that point, to allow the board to sit squarely. Take extreme care not to damage any traces in that area (see Figure 4).

Now tighten the screw completely and press the circuit board down onto the foam tape that is covering the plastic post. Your modem is now mounted and ready to be hooked up.

Figure 4. Circuit Board Notch Location
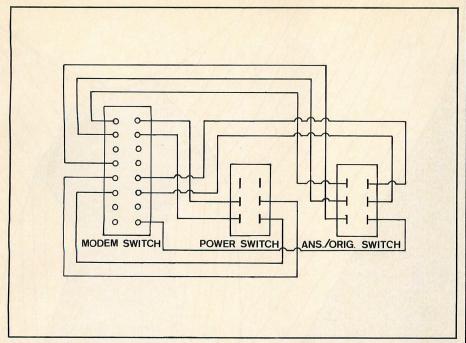
Figure 5. Switch to Modem Hook-up

Hooking Up
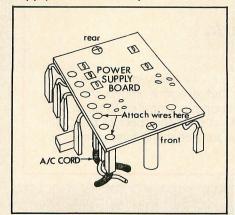
The four wires connecting the RS-232 will be connected first. Prepare the RS-232 jack in the computer in the same manner described above (adding solder to each terminal). The rightmost wire on the jack in the modem goes to the leftmost terminal on the jack in the computer. The wire second from the right in the modem goes to the terminal second from the left in the computer, and so on.

Place the case containing the switches and LEDs next to the computer so the wires will reach the modem board. Solder the wires from the LED on the left of the computer case to the two pins from the original LED on the modem. Be sure to observe that the polarity of the LED in the computer case matches the polarity printed next to the LED pins on the modem board. Information concerning the polarity of the LEDs you have installed in the computer case should be included on the package. No damage will result if they are installed backwards; they simply won't work, and you will have to reverse the leads. Be sure to insulate the connections on the modem board with electrical tape to prevent shorting. Refer to Figure 5 to hook up the switches to the modem board.

Now you need only connect the modem to the telephone lines and power source. The transformer is small enough to fit inside the computer next to the power supply. Solder an eight-inch piece of wire to each prong on the transformer and insulate with electrical tape. Solder the other ends to the power supply board, as shown in Figure 6. Using electrical tape, secure the transformer to the black metal "fingers" of the heatsink next to the power supply. Coil up the excess cable and tape it, leaving enough cable to reach the power jack on the modem board. If you prefer, the transformer may be left external and the power cable may be run into the computer case through a small notch cut into the back of the case. Obtain a modular telephone extension cord and plug one end into the telephone jack on the modem board. Make a small notch in the back of the computer case and run the extension out the back of the case. Mark this cable "to telephone jack." Run the phone cable, already attached to the modem, through the notch as well. Mark this cable "to telephone." ∎ ∎ ∎

Figure 6. Transformer to Power Supply Board Hook-up

GRAY FLANNEL

A few thoughts about the Color Computer in business.

by William Barden, Jr.

THE COLOR COMPUTER — a business machine? Why not? Yes, you say, but everyone *knows* the Color Computer is not a business computer! Well, as they used to say at B,B,D, & O, let's run the specs of the Color Computer up the flagpole and see if any businessman salutes it...

What *are* the requirements of a business computer? They're changing rapidly. Ten years ago, small businesses would have killed for a system as versatile as a disk-based Color Computer to run accounting packages, inventory, and reports. Ten years from now, nobody will be able to understand how businesses were able to suffer along with the antiquated IBM PC.

At this point in time, a "business system" might be loosely defined as having these qualities:

- A large quantity of business applications software
- Lots of disk space
- A moderately high-speed printer for reports
- A good keyboard and display
- User friendliness

Does the Color Computer meet these qualifications?

Leaving the question of business applications software aside for the moment, look at the other criteria. The Color Computer has a moderate amount of disk space. You could, for example, add two drives and be up to 320K bytes of disk storage, and that's without having to use any disk space for the operating system, as on some systems. Add two more drives, and you've got 640K bytes. No hard disk is available as yet, but you can store a lot of data even with two drives.

Now that Radio Shack makes their printers selectable for either parallel or serial interfacing, you can get some very nice printers for use on the Color Computer. A good example is the DMP-120 at 120 characters per second — a bargain at about $500. Need 132 columns for those massive business reports? You've got the DMP-420 at about $1000. Certainly there's no lack of any grade of printer for the Color Computer from Radio Shack — or add someone else's serial printer.

The new Color Computer keyboard is not as good as an IBM design, by any means, but it's perfectly adequate for extended typing. The display would admittedly be better at 64 or 80 characters per line, but 32 characters per line is workable.

As far as user friendliness goes, how can an office clerk be intimidated by the Color Computer? I'd certainly think the massive (in comparison) IBM PC would be much more imposing.

This brings us to business applications software for the Color Computer. It's one of those chicken/egg situations. The Color Computer has the user base to support more business packages, but software developers don't perceive it as a business system. Because there isn't much in the way of business software, nobody considers using it as a business system.

Could the Color Computer function as a business system if there were more applications software? It's certainly a state-of-the-art system with its 6809 microprocessor and the powerful commands of Extended Basic. The disk storage, keyboard, and display are workable as well. When you consider that there are a significant number of people in Great Britain using the Sinclair ZX-81 (the British equivalent of the Timex//Sinclair TS1000 you see advertised for $29.95) as a serious business system(!), I think it's safe to say the Color Computer is underrated as a potentially good system for small businesses! This especially applies to Mom and Pop operations that don't require and can't afford a Tandy 2000 or IBM PC with printer for $5500. They can get a system that'll do a fine job at less than half the price — with high-speed printer and disk storage.

Think about it in terms of what constitutes a "business system" currently. You may not need that Model 16 or IBM PC for your small business application after all! And you'll still be able to sneak in sessions of "The Sands of Egypt" after quitting time! ∎ ∎ ∎

Patching Scripsit

by Ashok Basargekar

R obert Lake's article in the August, 1983 issue of **The Color Computer Magazine** prompted me to prepare a patch to Scripsit on the RAM for use with my Line Printer VIII.

The Line Printer VIII can run at 1200 baud and I thought it would be nice if Scripsit would let me use this baud with my printer. At first I tried to POKE 150,41 before executing Scripsit, but this did not work. I also tried to add LDX #$0029 and STX $95 in Mr. Lake's patch to Scripsit. This also did not work. Finally after careful study of Scripsit, I developed the following patch that permits me to use nine different printer bauds. I have also added a feature that will let me underline any word in the text by enclosing it in the less than and greater than signs. Of course, since these signs are reserved by my patch for the underlining feature, they will not be available for their normal use.

As soon as you EXEC &H6000, a menu for selecting a baud appears on your screen. After the baud rate is selected, the main menu will appear on the screen.

To patch Scripsit with my features, you will need to create the object code of my assembly language program on tape, placed just after Scripsit in RAM. Prepare it following the instructions in Mr. Lake's article. To load the new Scripsit, you simply type CLOADM "SCRIPSIT" and then CLOADM my patch. Some of the original Scripsit's code will be overwritten, and some new codes will be added to Scripsit.

The entire new program can then be CSAVEd on tape: CSAVEM "SCRIPSIT", &H6000, &H7A16, &H6000.

I have tried to add enough comments in my patch to clarify the program. The assembly language program is written using EDTASM+ (Radio Shack). ■ ■ ■

Listing begins page 40

Charley Freiberg

Scripsit Redone

by Ralph Tenny

C OLOR SCRIPSIT IS a fairly good word processing package. The program pak version is excellent, except that it uses cassette for storage and lacks some features the disk version has. The disk version unhappily lacks two important features found in the cassette version. Even so, it is a good buy for all except the professional writer who demands high productivity. Most important, all versions of Scripsit are comfortable to use and are easily learned. So, let's avoid some of the shortcomings and improve our operating speed.

"S it back and watch the marvelous Color Computer do the rest. "

Document Formats

Formatting a document is most time-consuming. Business letters have inside addresses and other special arrangements. Each of my Disk Scripsit disks contains samples of various types of documents saved on the disk as an example file. By loading this file instead of starting with an empty file, I can quickly edit the date, address and salutation, then proceed to the letter. Since Scripsit writes over text my new letter soon "eats up" the original document, and I have been able to begin on the body of the letter almost immediately, instead of formatting a new document. The same can be done with cassette Scripsit by saving a short example of each format on a "read only" cassette.

Cassette Scripsit automatically saves the basic print parameters — text width, margins, lines per page, etc. — as part of the tape file, making document reprinting or editing really easy. Unfortunately, Disk Scripsit omitted this excellent feature, so the only possible remedy seems to be to make a cryptic entry in the sample file prompting you to re-enter these parameters in the menu. Disk Scripsit has a limited set of special print emphasis codes built into the print controls menu (option 7); these can be changed to suit your printer. For example, I substituted Epson "start italics" and "stop italics" codes for Radio Shack "start underline" and "stop underline" and "enhanced print" for "start elongation" codes. These new codes remained a permanent record on the disk, so they do not have to be entered each session.

It has been my experience that Color Scripsit's built-in paging commands do not result in predictable performance on the Epson printer, so I use one of two tricks. The first is to make each page a separate document that stands alone during the print process (and can be printed by the print spooling option of Disk Scripsit while other pages are entered and edited). The second option is to format the document completely on the screen and use the BREAK4 command to print the document, one line at a time. Of course, it gets tiring pressing the Break and numeral 4 keys during an entire page, but what else can you be doing during the print operation if you don't use spooling?

Headers and Footnotes

Although Color Scripsit does not have a satisfactory provision for headers, footnotes and similar special formats, these can be handled one of two ways. Make a trial print-out to determine where pages should end, and edit in the required header or footnote. If you are printing on single pages (with a friction feed printer, for example), you can prepare each page in advance with the required header, then print it as discussed above. Otherwise, simply make the header part of the page during the final edit process.

With mailing labels available in pin-feed format, it is easy to handle them by entering the required names and addresses in some sort of special format and use a Basic program to print the labels. For example, I use this data format:

%Joe Blow/3768 Windy Ave./Tempest NX 99999

By building the data base (the list of names) with slashes and percentage signs as field delimiters, it then becomes easy to write simple Basic programs to print the names in one of two formats. The first format is standard label format:

Joe Blow
3768 Windy Ave.
Tempest NX 99999

Because I maintain membership information (almost 300 names) for a national hobby organization, I also need to be able to furnish membership lists using this format:

Joe Blow, 3768 Windy Ave., Tempest NX 99999

After some experimentation I was able to develop Program Listings 1 and 2. They are simple, single-purpose programs to list data from only one file. They work for me; if I had several files to maintain, it would be expedient to make them interactive so they would prompt for a file name, number of copies of each label, number of names, etc. ■ ■ ■

Program Listing. Basargekar's Patch

32K Color Basic

```
798F  0173         00570         FDB   $0173          = 150
7991  00BE         00580         FDB   $00BE          = 300
7993  0057         00590         FDB   $0057          = 600
7995  0029         00600         FDB   $0029          = 1200
7997  0012         00610         FDB   $0012          = 2400
7999  0006         00620         FDB   $0006          = 4800
799B  0001         00630         FDB   $0001          = 9600
799D  BD A928      00640  BEGIN  JSR   CLS            CLEAR SCREEN
79A0  30 8D FF77   00650         LEAX  MES,PCR        GET MESSAGE POINTER
79A4  8D 16        00660         BSR   PRINT          SEND MSG TO SCREEN
79A6  DC 88        00670         LDD   <$88           GET CURSOR POINTER
79A8  ED 8C DD     00680         STD   STOPER,PCR     STORE AS STOPPER
79AB  8D 19        00690         BSR   SPECS          GO GET USER INPUT
                   00700  * THIS ROUTINE PICKS UP THE USER'S CHOICE
                   00710  * OF PRINTER BAUD RATE AND STORES IT AT $95
79AD  31 8C DB     00720         LEAY  BAUD,PCR
79B0  AE 8C A1     00730  LOOP   LDX   ,Y++
79B2  6A 8C D5     00740         DEC   TMP,PCR
79B5  26 F9        00750         BNE   LOOP
79B7  9F 95        00760         STX   <$95
79B9  16 EC0A      00770         LBRA  $65C6
                   00790  * PRINT MESSAGE ON SCREEN
79BC  A6 80        00800  PRINT  LDA   ,X+
79BE  27 05        00810         BEQ   DONE
79C0  BD A30A      00820         JSR   SCREEN
79C3  20 F7        00830         BRA   PRINT
79C5  39           00850  DONE   RTS
                   00860
```

TAPE LOADER

```
A928          00070  CLS    EQU   $A928
A30A          00080  SCREEN EQU   $A30A
A1C1          00090  INKEY  EQU   $A1C1
              00100         ORG   28142
6DEE  16 F841 00110         LBRA  $6632           MOVE FROM HERE TO ADD
6DFC  58      00120         ORG   $6DFC           UNDERLINE ROUTINE
6DFC          00130         ASLB                  THIS CHANGE ALLOWS
6E0F  7E A300 00140         ORG   $6E0F           NEW NUMBERS IN ADDR.
              00150         JMP   >$A300          149 & 150 FOR
              00160         ORG   $6632           NEW PRINTER BAUD RATE
6632  E4      00170         LDA   ,S              GET CHAR.
6634  81 3C   00180         CMPA  #$3C            IS IT ASCII FOR <?
6636  27 0B   00190         BEQ   UNDER           YES? GET CODE FOR UNDERLINE
6638  81 3E   00200         CMPA  #$3E            IS IT ASCII FOR >?
663A  26 02   00210         BNE   RETURN
663C  86 0E   00220         LDA   #$0E            GET END OF UNDERLINE CODE
663E  C6 08   00230  RETURN LDB   #8T             SET BIT COUNT
6640  16 07AF 00240         LBRA  28146           AND RETURN
6643  86 0F   00250  UNDER  LDA   #$0F            GET START OF UNDERLINE CODE
6645  20 F7   00260         BRA   RETURN
              00270         ORG   $6000
6000  16 1915 00280         LBRA  START
7918  16 0082 00300  START  LBRA  BEGIN
791B  5052494E5445  00310  MES    FCC  'PRINTER BAUD RATES:'
      52204241554E
      20524154553A                (Note: All FCC instruction output
                                   is artificially compressed. —Ed.)
792E  0D0D    00320         FDB   $0D0D
```

```
00870 * GET USER'S CHOICE OF BAUD RATE
00880 79C6 8D 34      SPECS  BSR   FLASH        FLASHING CURSOR
00890 79C8 81 31             CMPA  #$31         IS INPUT <1?
00900 79CA 25 FA             BLO   SPECS        IF SO, TRY AGAIN
00910 79CC 81 39             CMPA  #$39         IS INPUT >9?
00920 79CE 22 F6             BHI   SPECS        IF SO, TRY AGAIN
00930 79D0 BD A30A           JSR   SCREEN       SEND INPUT TO SCREEN
00940 79D3 80 30             SUBA  #$30         CONVERT TO DECIMAL
00950 79D5 A7 8C B2          STA   TMP,PCR
00960 79D8 8D 22      LP     BSR   FLASH
00970
00980 * GIVE USER A CHANCE TO CHANGE SELECTED
00990 * BAUD RATE WITH LEFT ARROW
01000 79DA 81 08             CMPA  #$08         LEFT ARROW PRESSED?
01010 79DC 26 19             BNE   LP3          IF NOT
01020
01030 * DON'T ERASE THE MESSAGE
01040 79DE DC 88      LP1    LDD   <$88
01050 79E0 10A3 8C 04        CMPD  STOPER,PCR
01060 79E4 24 89             BHS   LP2
01070 79E6 0C 89             INC   <$89
01080 79E8 20 F4             BRA   LP1
01090
01100 * MOVE CURSOR TO THE LEFT
01110 79EA 86 8F      LP2    LDA   #$8F
01120 79EC A7 9F 0088        STA   [$88]
01130 79F0 86 08             LDA   #$08
01140 79F2 BD A30A           JSR   SCREEN
01150 79F5 20 CF             BRA   SPECS        GO BACK FOR INPUT
01160 79F7 81 0D      LP3    CMPA  #$0D         ENTER KEY?
01170 79F9 26 DD             BNE   LP           IF NOT
01180 79FB 39                RTS
01190
01200 * CURSOR ROUTINE
```

```
00330 7930 313A20202037   FCC   '1:  75'
      7937 35
00340 7938 323A20203131   FCB   $0D
00350      30             FCC   '2: 110'
00360 793F 0D             FCB   $0D
00370 7940 333A20203135   FCC   '3: 150'
      7947 30
00380 7947 0D             FCB   $0D
00390 7948 343A20203330   FCC   '4: 300'
      794F 30
00400 794F 0D             FCB   $0D
00410 7950 353A20203630   FCC   '5: 600'
      7957 30
00420 7957 0D             FCB   $0D
00430 7958 363A20313230   FCC   '6: 1200'
      795F 30
00440 795F 0D             FCB   $0D
00450 7960 373A20323430   FCC   '7: 2400'
      7967 30
00460 7967 0D             FCB   $0D
00470 7968 383A20343830   FCC   '8: 4800'
      796F 30
00480 796F 0D             FCB   $0D
00490 7970 393A20393630   FCC   '9: 9600'
      7977 30
00500 7977 0D0D           FDB   $0D0D
00510 7979 656E74657220   FCC   'enter 1 TO 9: '
           3120544F2039
           3A20
00520 7987 00             FCB   $00
00530 7988 0000     STOPER FDB  $0            CURSOR POINTER
00540 798A 00       TMP    FCB  $0            TEMP. FOR USER INPUT
00550 798B 02EB     BAUD   FDB  $02EB         = 75
00560 798D 01CA            FDB  $01CA         = 110
```

```
79FC 86    80        01210 FLASH  LDA    #$80
79FE 8D    09        01220        BSR    KBSCAN
7A00 26    06        01230        BNE    OUT
7A02 86    8F        01240        LDA    #$8F
7A04 8D    03        01250        BSR    KBSCAN
7A06 27    F4        01260        BEQ    FLASH
7A08 39              01270 OUT    RTS
7A09 A7    9F 0088   01280 KBSCAN STA    [$88]
7A0D BD    A1C1      01290 SCAN   JSR    INKEY
7A10 26    03        01300        BNE    OUT1
7A12 5A              01310        DECB
7A13 26    F8        01320        BNE    SCAN
7A15 39              01330 OUT1   RTS
                     01340
          7918       01350        END    START
00000 TOTAL ERRORS

BAUD    798B
BEGIN   799D
CLS     A928
DONE    79C5
FLASH   79FC
INKEY   A1C1
KBSCAN  7A09
LOOP    79B0
LP      79D8
LP1     79DE
LP2     79EA
LP3     79F7
MES     791B
OUT     7A08
OUT1    7A15
PRINT   79BC
RETURN  663E
SCAN    7A0D
SCREEN  A30A
SPECS   79C6
START   7918
STOPER  7988
TMP     798A
UNDER   6643
```

TAPE LOADER

32K Disk Extended Color Basic

Program Listing.

Tenny's Membership List

```
10 CLEAR2000
20 GOTO1000
30 L1$=""
35 FOR P=1TO100 '100 CHARS. MAX
PER DATA ENTRY
40 CH$=MID$(A$,P,1)
50 IF CH$="/" THEN200
60 IF CH$="" THEN150
70 L1$=L1$+CH$
80 NEXT P
150 PRINT#-2,L1$:RETURN
200 CH$=", ":L1$=L1$+CH$
210 NEXT P
1000 OPEN"I",#1,"NIMLIN/TXT"
1005 L=1 'LINE COUNTER FOR LABEL
  FORMAT
1010 FOR C=1TO300 'READ WHOLE FI
LE
1020 LINEINPUT#1,A$ 'ONE LINE AT
 A TIME
1025 IF A$="" THEN2000 'END OF F
ILE
1030 GOSUB30
1032 L=L+1:IF L>56 THEN2010 'PAG
E FORMATTING
```

```
1035 NEXT C
1037 A$=MID$(A$,2) 'SKIP "%"
2000 CLOSE 1:STOP 'ALL PRINTED
2010 PRINT#-2,CHR$(12); 'FORM FE
ED
2015 L=1:GOTO1035 'NEW LINE COUN
T FOR NEXT NAME ■■■
```
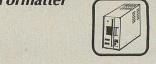
TAPE LOADER

32K Disk Extended Color Basic

Program Listing.

Tenny's Label Formatter

```
5 CLEAR2000
10 GOTO1000
20 L1$="":L2$="":L3$="":L4$="":L
5$=""
30 P=2
40 CH$=MID$(A$,P,1)
55 IF CH$="/"THEN70
60 L1$=L1$+CH$
63 P=P+1
66 GOTO70
70 P=P+1:CH$=MID$(A$,P,1)
75 IF CH$="/" THEN90
80 L2$=L2$+CH$
85 GOTO70
90 P=P+1:CH$=MID$(A$,P,1)
95 IF CH$="/" THEN110
100 IF CH$="" THEN180
105 L3$=L3$+CH$:GOTO90
110 P=P+1:CH$=MID$(A$,P,1)
115 IF CH$="/" THEN150
120 IF CH$="" THEN180
130 L4$=L4$+CH$:GOTO110
150 P=P+1:CH$=MID$(A$,P,1)
155 IF CH$="" THEN180
160 L5$=L5$+CH$:GOTO150
180 RETURN
1000 OPEN"I",#1,"NIMLIN/TXT"
1010 FOR X=1TO300
1015 LINEINPUT #1,A$
1017 IF A$="" THEN1120
1020 GOSUB20
1025 Y=1
1030 PRINT#-2,L1$:Y=Y+1
1032 PRINT#-2,L2$:Y=Y+1
1034 PRINT#-2,L3$:Y=Y+1
1040 IF L4$="" THEN1100
1050 PRINT#-2,L4$:Y=Y+1
1060 IF L5$="" THEN1100
1070 PRINT#-2,L5$:Y=Y+1
1100 PRINT#-2,CHR$(13);:Y=Y+1
1115 NEXT X
1120 STOP ■■■
```

Circle No. 58 on Reader Service Card

TRS-80® Color Computer Even Better.

The Heart of a Professional Disk-Based System. Add a Color Computer disk drive for an extra 156,000 characters of storage and quicker access to your data. You can add up to four disk drives for a vast 626,000 characters of total storage. For advanced programmers, our new OS-9 operating system (26-3030, $69.95) allows you to utilize the full 64,000-character memory for assembly language programs. OS-9 includes a text editor, assembler and debugger. You can develop and edit your own assembly language programs, convert them into machine language for execution, then test them for errors.

Multi-Pak Interface

179⁹⁵
26-3024

USE YOUR

Radio Shack
CITILINE
YOUR CARD
CITIBANK©

Easily Expandable. Best of all, your 64K Color Computer 2 will grow with your future needs. Add our Deluxe Joysticks for more accurate cursor control and faster response with your favorite games. Or choose our Multi-Pak Interface, which lets you connect up to four Program Paks to your Color Computer at once. When you're ready to change programs, just move the selector switch. Go on-line with the world with our deluxe RS-232 Program Pak (26-2226, $79.95). You can communicate with national information services and local bulletin boards by telephone—just add a modem. You can also add a printer, a Color Mouse cursor controller and more!

See It Today. The 64K Extended BASIC Color Computer 2 is available today at over 1000 Radio Shack Computer Centers and at participating Radio Shack stores and dealers nationwide.

Disk Drive #0 Kit

399⁹⁵
26-3029

AS LOW AS

Radio Shack
CITILINE
YOUR CARD
CITIBANK©

$28 PER MONTH

Send me the all-new 1984 TRS-80 catalog RSC-11.

Mail To: Radio Shack, Dept. 84-A-619
300 One Tandy Center, Fort Worth, Texas 76102

NAME _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

TELEPHONE _____

Radio Shack®

The biggest name in little computers®
A DIVISION OF TANDY CORPORATION

STATISTICS: A TUTORIAL

by David Lionell Dawson

Keith Shortall

BEFORE I JOINED the University of Nebraska staff, I recall skipping over statistical programs in computer magazines in my rush to the game or utility programs. This was really unfortunate; many statistics programs can prove useful in personal and business applications, once the user understands a few basic ideas. I will attempt to show you how to use these programs, and discuss statistics in general.

Descriptive Statistics

All statistical procedures can be loosely categorized as *inferential* or *descriptive*. Descriptive statistics describe, or summarize, a group of numbers. Program Listing 1 is my package for doing a group of descriptive statistics on any set of scores. The scores could be almost anything — from sales during the week, to the amount of your phone bills. In the sample output shown in Table 1, the program is analyzing the cost of various Color Computer software packages. I wanted to know whether a particular $39 program was relatively expensive or cheap.

The first set of printed values attempt to find the most typical or average score, or price, in this case. The first of these is the *mean* which is what most people would call the "average." The mean is just the scores added together and divided by the number of scores you are considering. This is a very common descriptive statistic, and is probably familiar to most readers. The mean demonstrates how descriptive statistics describe or summarize — by taking a large group of numbers and turning this group into one score that is representative of them all.

The next value, the *median*, is simply the exact middle score. A formula is used in the program that considers even numbers of scores and tied scores at the middle of the group when the mean calculation is made. This is why the score may appear as a fraction, even when only whole numbers were under consideration. The idea is the same: the median is the middle score.

The median and mean are both important because the mean is affected more by the value of *each* score than is the median. Often the mean is not the best, or most typical, average score. The median is commonly used, for example, when figures are given stating the average income in the United States. To demonstrate why this is done, suppose that only three people live in the U.S., and that they make $10,000, $12,000, and $14,000. The mean here would be $12,000, and it would be a good estimate of the average.

Summary Statistics
Software for Color Computer

	CENTRAL VALUE		VARIABILITY	
MEAN	33.9515001	RANGE	85
MEDIAN	24.7	STANDARD DEVIATION	23.5893629
MODE	21.95	SKEWNESS	1.65374558
		KURTOSIS	1.45556159

Software for Color Computer Frequency Distribution

SCORES	FREQ	CUMFREQ	CUMPERCENT
95 – 110	1	20	1
79 – 94	1	19	.95
63 – 78	1	18	.9
47 – 62	1	17	.85
31 – 46	1	16	.8
15 – 30	14	15	.75
-1 – 14	1	1	.05

Table 1. Output from Descriptive Statistics Program

Now suppose that the person earning $14,000 wrote and sold a great arcade game; the next year his earnings rose to $1,000,000. With salaries of $10,000, $12,000 and $1,000,000, the mean is now around $340,000 — not a very true picture of what the average person makes! The median, or middle score, would still be $12,000, because it is still the score between the lowest and highest.

The last measure of average score is the *mode*, or merely the score that occurs the most. If you look at Line 10010 of Listing 1, you will find four scores of $21.95. No other score appears that often, which makes it the mode.

The next two scores have to do with variability, which is how different from each other the numbers we are considering are. Does everyone get a grade of 100 on the test, or is there a wide range of scores?

The first statistical measure of the variability of scores is the *range*. This is merely the high score minus the low score. The *range is an inferior measure*; for example, one very high score on one test in a history class would make a student seem very inconsistent, while all her other test scores might be very close together.

The *standard deviation* is a much better and more widely accepted measure of variability. This value would be large if, looking for instance at the sales made by your salespeople during a given month, a few sales were very high and a few were very low. The value would be small if all your sales staff did about equally. The standard deviation represents how stretched out the scores are, or how much they differ from each other.

The last two values in the summary statistics portion refer to the shape of the group of scores. This idea will become clearer as we look at each in turn.

Skewness has to do with the number of high scores versus the number of low scores. If there are more high scores than low scores, the skewness would be a negative number; but, if there were more low scores than high, the number would be positive.

Kurtosis is how spread, or pointed, a group of scores are. The more the scores are spread evenly, the more negative the kurtosis score becomes, and the more the scores are the same, or pile up, the more positive the score becomes. Skewness and kurtosis statistics range from about −2.0 to +2.0.

The Frequency Distribution Table gives: the number of scores within a range (labeled FREQ); the number below that range (labeled CUMFREQ); and the percent of the total the CUMFREQ represents (labeled CUMPERCENT). Looking at Table 1, it could be said that 85 percent of the scores are below 53.

Using the Info

From the sample output in Table 1 we can see that the mean and median are very different. If we also look at Figure 1, we can see there are a few very high scores which, if you remember the earlier discussion, affect the mean more than the median.

Refer to Figure 1; it is easy to understand the skewness and kurtosis numbers. The skewness of 1.65 indicates more low scores than high, as is quite evident in the polygon. The kurtosis of 1.45 means a very peaked distribution, or that many of the scores are the same — also apparent in Figure 1.

So, what do these final figures mean? It appears that there are a large number of software packages between $20 and $30, but also a few costing as much as $100; the average is about $24 (using the median). My $39 program, therefore, is above average. Using the Frequency Distribution Table, I can see that about 80 percent of all packages included in my study cost less than $39. The software I was considering buying is more expensive than 80 percent of the packages advertised. I decided to look for a cheaper program.

Inferential Statistics

The second broad category of statistics is called *inferential statistics*. These are more complicated than descriptive statistics. Instead of trying to just summarize data, these procedures ask questions like: Is one group in a class really composed of better readers? Are more sales made by older or younger salespeople?

Though more complicated, the answers to this type of question seem more satisfying than those provided by descriptive statistics. This discussion of inferential statistics will be very limited: I will demonstrate what these procedures can accomplish, but I strongly recommend those who are not acquainted with these statistics do some background reading.

Suppose you were interested in whether the new policy you began at home — paying your children an allowance based on completion of chores — was increasing the amount of chores done around the house. You decide to test the idea; you get a measure of the number of chores done by each child for one of the days before you started your new system, and another measure for one of the days after the system took effect. How would you know if there was a real difference?
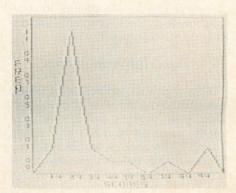

Photo 1. Anova Graph—Frequency Polygon

Using the Analysis of Variance Program (ANOVA) given in Listing 2, you could produce Table 2, an analysis that would answer this question. The columns SS, df (degrees of freedom), and MS refer to quantities beyond the scope of this discussion, but what is important is the number at the bottom labeled F. The F value can be looked up in a simple table found in any statistics book, and will tell us immediately if the groups are significantly different. To look up the value, use the df which appear to the left of the equals sign: in this case 1 and 3. The 1 is the numerator df and 3 is the denominator df. Find the 1 in the table column and the 3 in the table row (in most tables), and see where they meet. This will give you the critical F. If your F number is large enough, the groups are different. Since many of you do not have a table available to you, as a rule of thumb for degrees of freedom of moderate size, with a total greater than 10, an F of more than four will be significant. As df's become larger the necessary F becomes smaller.

In this example, where df=1,3, the necessary F is 10.13 for P=.05, or a 5 percent chance of being wrong (this percentage is taken from a statistics book). The value found of 2.45 (see Table 2), therefore, is too small for significance, which means the two groups are probably not different. In plain words, you should stop wasting money paying your children for chores, as it does not help.

ANOVA

The ANOVA program presented here allows for repeated measure. This is very rare, but it is much more useful than those programs that require the researcher to place different people in each group being studied. For example, in our allowance test, a non-repeated measures program would require half the chil-

MEANS
Mean for Subject 1 = 1
Mean for Subject 2 = 2
Mean for Subject 3 = 3.5
Mean for Subject 4 = 5
Mean for Subject 1 = 2.5
Mean for Subject 2 = 3.25

ANOVA Summary Table
Repeated Measures One Way

SOURCE	SS	df	MS
Between S	18.375	3	
Within S	2.5	4	
Treat	1.12500003	1	1.12500003
Residual	1.37499997	3	.458333323
Total	20.875	7	

F with degrees of freedom 1,3 = 2.45454557

Table 2. Output from ANOVA

```
ANOVA SUMMARY REPEATED ONE WAY

SOURCE    SS      DF     MS

BETW.  18.375    3
WITHIN 2.5       4

TREAT  1.12500003  1   1.12500003

RESID. 1.37499997  3  .458333325

TOTAL  20.875     7

F ( 1 , 3 )= 2.45454557
OK
```

```
INSTRUCTIONS (Y/N)? N
KEY (I) FOR KEYBOARD INPUT OF
DATA OR (D) FOR DATA STATEMENTS
? D
PRINTER OUTPUT (Y/N)? N
MEANS (Y/N)? Y
          MEANS
MEAN SUBJECT  1 = 1
MEAN SUBJECT  2 = 2
MEAN SUBJECT  3 = 3.5
MEAN SUBJECT  4 = 5
MEAN LEVEL    1 = 2.5
MEAN LEVEL    2 = 3.25
KEY ENTER FOR SUMMARY TABLE?
```

Photo 2a. and 2b. Data Screens

dren to get an allowance for doing chores, and half to be given allowance money and not do chores. This is clearly impractical, as it would be in a classroom, where it is hard to treat two separate groups differently. With ANOVA this difficulty needn't arise.

The type of data that can be used with this program is quite limited, as are nearly all microcomputer ANOVA programs. If data is collected on the same people in several levels of some condition, this analysis will be appropriate most of the time. Any F larger than 4.0 will be significant, but unless the user has some statistical knowledge the *accuracy* of the result cannot be considered complete. Many topics I have had to skip here are necessary to accurately determine the appropriateness of this analysis for a specific type of data.

Using the Programs

The most important detail when using these programs is to enter the kind of data the program expects, in the expected order. The Descriptive Statistics Program can be used with almost any type of numbers, unlike the ANOVA program. However, the Frequency Distribution Table (Table 1) and graph assume integer or mixed numbers, not all decimal fractions. Multiply all data by a constant if you find yourself with decimal fractions, so answers will be displayed correctly.

The Frequency Distribution Table prints the number of scores within the interval .5 above to .5 below the interval printed — a convention. The graph is similar, and counts and plots the number of scores within the interval represented by its midpoint at the bottom. Most importantly, the graph plots the last two whole numbers in the scores as the horizontal scale, so if you use decimal fractions all the numbers on the horizontal will be zero. Also, if you use very large numbers the horizontal will be difficult to interpret. Use the graph only to get a feeling for the general distribution of the scores, and use the table for precise numbers within specific ranges.

The most obvious and potentially useful modification would be the capability to save data files to disk or tape. A routine to input from tape or disk could be added to Listing 1 after Line 50. The variable N holds the number of data points, and the array D(N) holds the data. Be sure to dimension D in the routine.

The same kind of routine could be added after Line 110 in Program Listing 2. The number of levels is stored in K, the number of subjects in N, and the data in D(K,N). This routine should also contain a dimensioning of the variable D.

A Save routine could be constructed appearing after Line 140 in Listing 1 and Line 180 in Listing 2. If the routines were inserted as suggested, data could still be edited in data statements and then saved to tape or disk.

Statistics can be used in many situations. The first program should be accessible to everyone, while the second should be used with caution — it is the most powerful, but there are many considerations that have not been covered here. However, feel free to experiment with the program; it is a tool used by many scientists and should at least provide insight into scientific methods. I hope my introduction has shown the potential of inferential statistics and will motivate you to continue reading about the subject. ■ ■ ■

16K Extended Color Basic

Program Listing 1. Descriptive Statistics

```
6 REM REMOVE REMARKS TO PROVIDE
MORE ROOM FOR DATA IF YOU WISH.
10 PCLEAR4:CLS:PRINT STRING$(32,
"#")
20 PRINT:PRINT"     DESCRIPTIVE
STATISTICS":PRINT:PRINT"      BY
DAVID LIONELL DAWSON"
30 PRINT:PRINTSTRING$(32,"#"):PR
INT:PRINT"KEY ENTER TO CONTINUE
40 IF INKEY$="" THEN 40
45 REM OPTIONS.
50 CLS:PRINT@192,"(K) EYBOARD EN
TRY OR (D) ATA     STATEMENTS";:I
NPUT CH$
60 IF CH$="D" THEN 110
65 REM KEYBOARD DATA ENTRY.
70 INPUT"NUMBER OF CASES";N
80 DIM D(N), F(N)
90 FOR X=1 TO N:PRINT "ENTER CAS
E #";X;:INPUT D(X):NEXT
100 GOTO 140
110 CLS:PRINT:PRINT:PRINT"BEGIN
DATA IN LINE 10000 GIVE    NUMBER
 OF CASES THEN LIST THE    DATA I
N ANY ORDER.":PRINT"IF ALL THE D
ATA IS ENTERED KEY  'R' ELSE PRE
SS ANY OTHER KEY ANDTHE PROGRAM
WILL STOP."
120 N$=INKEY$:IF N$="" THEN 120
ELSE IF N$<>"R" THEN END
125 REM DATA STATEMENT DATA ENTR
Y.
130 READ N:DIM D(N):FOR X=1 TO N
:READ D(X):NEXT
140 FOR X=1 TO N:T=T+D(X):NEXT:M
=T/N
150 PRINT"COMPUTING ... PLEASE W
AIT ...";
155 REM SORT ROUTINE.
160 FOR X=1 TO N:PRINT"...";:FOR
 Y=1 TO N:IF D(X)<D(Y) THEN W=D(
Y):D(Y)=D(X):D(X)=W
170 NEXTY,X
175 REM CALCULATE MEDIAN.
180 MI=INT(N*.5)
185 IF D(MI)<>D(MI+1) AND N/2=IN
T(N/2) THEN MD=(D(MI)+D(MI+1))/2
:GOTO260
190 LL=D(MI)-.5
192 FOR X=MI TO 0 STEP -1:IF D(X
)=D(MI) THEN NEXT
200 NB=X:CC=MI
210 CC=CC+1:IF CC<=N THEN IF D(M
I)=D(CC) THEN CX=CX+1:GOTO210
220 C2=MI
230 C2=C2-1:IF C2>0THEN IF D(MI)
=D(C2) THEN CZ=CZ+1:GOTO230
240 NW=CX+CZ+1
250 MD=LL+((N/2-NB)/NW)
255 REM CALCULATE MODE.
260 CO=1:W=0:FOR X=1 TO N
270 CO=X+1
280 IF CO<N+1 THEN IF D(X)=D(CO)
 THEN IW=IW+1:CO=CO+1:GOTO280
290 IF IW>W THEN W=IW:MO=D(X)
300 IW=0:NEXT X
305 REM CALCULATE STANDARD DEVIA
TION.
310 T=0:PRINT"***";:FOR X=1 TO N
:T=T+(D(X)-M)^2:NEXT:SD=SQR(T/N)
315 REM CALCULATE SKEWNESS.
320 T=0:PRINT"*";:FOR X=1 TO N:T
=T+((D(X)-M)/SD)^3:NEXT:SK=T/N
325 REM CALCULATE KURTOSIS.
```

```
330 T=0:PRINT"*";:FOR X=1 TO N:T
=T+((D(X)-M)/SD)^4:NEXT:KU=T/N-3
335 REM CALCULATE RANGE.
340 RA=D(N)-D(1)
345 REM DISPLAY STATISTICS.
350 CLS:PRINT@192,"SUMMARY STATI
STICS READY.":PRINT:INPUT"OUTPUT
 TO (P) RINTER OR          (S) CR
EEN";Y$:IF Y$="P" THEN 420
360 CLS:PRINT:PRINT:PRINT"SUMMAR
Y STATISTICS":PRINT:PRINT"CENTRA
L TENDENCY":PRINT"MEAN";M:PRINT"
MEDIAN";MD:PRINT"MODE";MO
370 PRINT:PRINT"VARIABILITY":PRI
NT"RANGE";RA:PRINT"STANDARD DEVI
ATION";SD:PRINT
380 PRINT"SKEWNESS";SK:PRINT"KUR
TOSIS";KU
390 PRINT:PRINT"KEY TO CONTINUE"
400 IF INKEY$="" THEN 400
410 GOTO 480
415 REM PRINT SUMMARY STATISTICS
.
420 INPUT"NAME OF DATA";N$:PRINT
#-2,TAB(31)"SUMMARY STATISTICS":
PRINT#-2,"":PRINT#-2,"":PRINT#-2
,N$
430 PRINT#-2,TAB(33)"CENTRAL VAL
UE":PRINT#-2,"":PRINT#-2,"MEAN";
M
440 PRINT#-2,"MEDIAN";MD:PRINT#-
2,"MODE";MO
450 PRINT#-2,"":PRINT#-2,TAB(34)
"VARIABILITY":PRINT#-2,""
460 PRINT#-2,"RANGE";RA:PRINT#-2
,"STANDARD DEVIATION";SD:PRINT#-
2,""
470 PRINT#-2,"SKEWNESS";SK:PRINT
#-2,"KURTOSIS";KU
475 REM CONSTRUCTION OF FREQUENC
Y DISTRIBUTION TABLE.
480 CLS:PRINT@192,"CONSTRUCT FRE
QUENCY DISTRIBUTIONTABLE (Y/N)";
:INPUT Y$:IF Y$="N" THEN 640
490 INPUT "NUMBER OF INTERVALS";
NO:IF NO=0 THEN IF N>15 THEN NO=
15 ELSE NO=N
500 NO=INT((((D(N)+1)/NO)+.5)+.5
)
510 INPUT"OUTPUT TO (S) CREEN OR
          (P) RINTER";Y$:IF Y$="
P" THEN 600
520 PRINT"SCORES    FREQ    CUMFRE
Q   CUM%
530 FOR X=INT(D(N)+.5) TO D(1)-N
O STEP -NO:PRINTUSING "###";X;:P
RINT"-";:PRINTUSING"###";X+NO-1;
:T=0:FOR Z=1 TO N:IF D(Z)>X-.5 A
ND D(Z)<X+NO-1+.5 THEN T=T+1
540 NEXTZ:PRINT"    ";T;
550 PRINT"      ";N-CF;"      ";:IF
CF<=0 THEN PRINT"1.00":CF=T+CF E
LSE PRINTUSING"#.###";(N-CF)/N:C
F=T+CF
560 NEXT X
570 PRINT"KEY ENTER"
580 IF INKEY$="" THEN 580
590 GOTO 640
595 REM PRINT DISTRIBUTION TABLE
 TO PRINTER.
600 INPUT"NAME";N$:PRINT#-2,N$:P
RINT#-2,"":PRINT#-2,"FREQUENCY D
ISTRIBUTION":PRINT#-2,STRING$(80
,"-"):PRINT#-2,"SCORES";TAB(20)"
FREQ";TAB(40)"CUMFREQ.";TAB(60)"
CUMPERCENT"
610 FOR X=INT(D(N)+.5) TO D(1)-N
O STEP -NO:PRINT#-2,X;"-";X+NO-1
;:T=0:FOR Z=1 TO N:IF D(Z)>X-.5
AND D(Z)<X+NO-1+.5 THEN T=T+1
620 NEXT Z:PRINT#-2,TAB(20)T;:PR
INT#-2,TAB(40)N-CF;:IF CF=0 THEN
 PRINT#-2,TAB(60)"1.00":CF=T+CF
ELSE PRINT#-2,TAB(60)(N-CF)/N:CF
=T+CF
630 NEXT X
635 REM DRAW POLYGON.
640 CLS:PRINT:PRINT:PRINT"POLYGO
N MENU":PRINT@192,"(F) OUR COLOR
 OR (T) WO COLOR";:INPUT Y$:IF Y
$="F" THEN PMODE3,1 ELSE PMODE 4
,1
650 IF Y$="F" THEN INPUT"COLOR O
F POLYGON (6)=CYAN, (7)= MAGENTA
, (8)=ORANGE";C:COLORC,5
660 IF Y$="T" THEN INPUT "COLOR
OF POLYGON (0)=BLACK, (5)=BUFF";
C:IF C=0 THEN COLOR 0,5 ELSE COL
OR 5,0
665 REM CONSTRUCT DRAW STRINGS F
OR NUMBERS.
670 PCLS:DIM NU$(10):NU$(1)="BD1
ND6":NU$(2)="BD1NL3R3D3L6D3R6BU"
:NU$(3)="BD1NL3R3D3NL6D3L6":NU$(
4)="BM+3,+1D4L6NU4R6D2":NU$(5)="
BD1NR3L3D3R6D4L6U4":NU$(6)="BD1NR3
L3D3D3R6D4L6":NU$(7)="NL3R3D6":N
U$(8)="BD1R3D3NL6D3L6U6R3":NU$(9
)="R3D3ND3L6U3R3"
```

```
680 NU$(0)="BD1R3D6L6U6R3"
685 REM DRAW LABELS ON GRAPH.
690 LINE (20,0)-(255,176),PSET,B
:DRAW "S3BM4,48L8D4NR8D4BD4NR8D4
ND4R8NU4L2D4BD4L8D4NR8D4R8BD4L8D
8NE4NG4R8U8":DRAW"S3BM106,185NR8
D4R8D4NL8BR6NR8U8R8BR6R8D8L8U8BR
14ND8R8D4NL4D4BR6NR8U4NR8U4R8BR6
NR8D4R8D4L8
700 SCREEN1,1
705 REM CALCULATE FREQUENCY WITH
IN INTERVAL.
710 IF MO=0 THEN MO=10
720 X=20:ST=INT((D(N)-D(1))/8.2)
:IF ST<2 THEN ST=2
730 T=0:FOR Z=D(1) TO D(N) STEP
ST:T=0:FOR X=1 TO N:IF D(X)>Z-.5
 AND D(X)<Z+ST-1+.5 THEN T=T+1
740 NEXT X:IF T>FE THEN FE=T
750 NEXT Z
760 MH=(FE*24)+14:IF FE<6 THEN Z
=6 ELSE Z=FE
770 IF MH>160 THEN MH=160
775 REM LABEL VERTICAL SCALE.
780 LL=INT(Z/7+.5):FOR X=14 TO 1
58 STEP 24:W$=STR$(Z):DRAW"BM14,
"+STR$(X)+NU$(VAL(MID$(W$,LEN(W$
)-1,1))):DRAW"BM14,"+STR$(X+8)+N
U$(VAL(RIGHT$(W$,1))):Z=Z-LL:IF
Z<0 THEN Z=0
790 NEXT X
795 REM LABEL HORIZONTAL SCALE.
800 X=20:FOR Z=D(1) TO D(N) STEP
 ST:W=INT((Z+ST+Z)/2):X=X+23:IF
X>247 THEN NEXT Z ELSE W$=STR$(W
):DRAW"BM"+STR$(X)+",177"+NU$(VA
L(MID$(W$,LEN(W$)-1,1))):DRAW"BM
"+STR$(X+8)+",177"+NU$(VAL(RIGHT
$(W$,1))):NEXT Z
810 LINE(20,176)-(20,176),PSET
815 REM DRAW LINES.
820 D=20:FOR Z=D(1) TO D(N) STEP
 ST:T=0:FOR X=1 TO N:IF D(X)>Z-.
5 AND D(X)<Z+ST-.6 THEN T=T+1
830 NEXT X:D=D+24:LINE-(D,176-(M
H*(T/FE))),PSET:NEXTZ
840 LINE-(D+20,176),PSET
850 IF INKEY$="" THEN 850
860 INPUT "CONTINUE (Y/N)";Y$:IF
 Y$="Y" THEN RUN ELSE END
875 REM THE FOLLOWING IS SAMPLE
DATA (SEE TEXT).
885 REM DELETE THIS BEFORE SAVIN
G THE PROGRAM.
10000 DATA 20
10010 DATA 94.95,69.95,49.95,19.
98,24.95,21.95,21.95,14.95,21.95
,91.95,24.95,9.95,21.95,19.95,29
.95,24.95,19.95,29.95,24.95,39.9
5                          ■■■
```
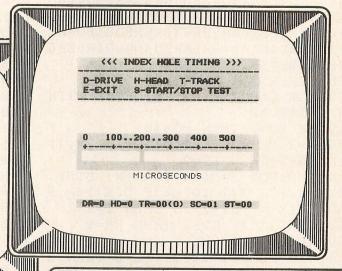

Program Listing 2. ANOVA

16K Extended
Color Basic

```
1 REM ONE-WAY REPEATED
6 REM FOR EXTRA SPACE FOR DATA D
ELETE ALL REMARKS AND USE THE CO
MMAND: POKE 25,6:NEW <ENTER>
9 REM BEFORE THE PROGRAM IS LOAD
ED.
10 CLS:PRINT@96,"      ANALYSIS
OF VARIANCE":PRINT:PRINT"FOR ONE
-WAY DESIGNS":PRINT"WITH REPEATE
D MEASURES.
20 PRINT:PRINT STRING$(32,128):P
RINT:PRINT"PROGRAM BY DAVID LION
ELL DAWSON BASED ON FORMULAS IN
WINER, B.J.STATISTICAL PRINCIPLE
S IN EXPER-IMENTAL DESIGN. NEW Y
ORK: MCGRAW-HILL, 1971."
30 IF INKEY$="" THEN 30
40 CLS:PRINT@160,"INSTRUCTIONS (
Y/N)";:INPUTY$:IF Y$="N" THEN 10
0
45 REM INSTRUCTIONS.
50 PRINT"THIS PROGRAM CALCULATES
 AN       ANALYSIS OF VARIANCE FO
R ONE     VARIABLE DESIGNS  WITH
EVERY     SUBJECT APPEARING IN EV
ERY LEVEL"
60 PRINT:PRINT"THERE MUST BE EQU
AL NUMBERS OF   OBSERVATIONS IN E
ACH CELL (ONE   FOR EACH SUBJECT)
 BUT THERE MAY BE ANY NUMBER OF
LEVELS OF THE   VARIABLE."
70 INPUT"(MORE)";XX
80 PRINT"YOU MAY USE EITHER DATA
 STATE-   MENTS OR KEYBOARD DATA
ENTRY.    IF DATA STATEMENTS ARE
USED      BEGIN WITH LINE NUMBER
5000."
90 PRINT"THE NUMBER OF LEVELS SH
OULD AP- PEAR FIRST THEN THE NUM
BER OF    SUBJECTS IN EACH THEN T
HE DATA   WITH ALL THOSE IN ONE L
EVEL TO-  GETHER IN THE SAME ORDE
R IN EACHLEVEL.":PRINT:PRINT
95 REM OPTIONS.
```
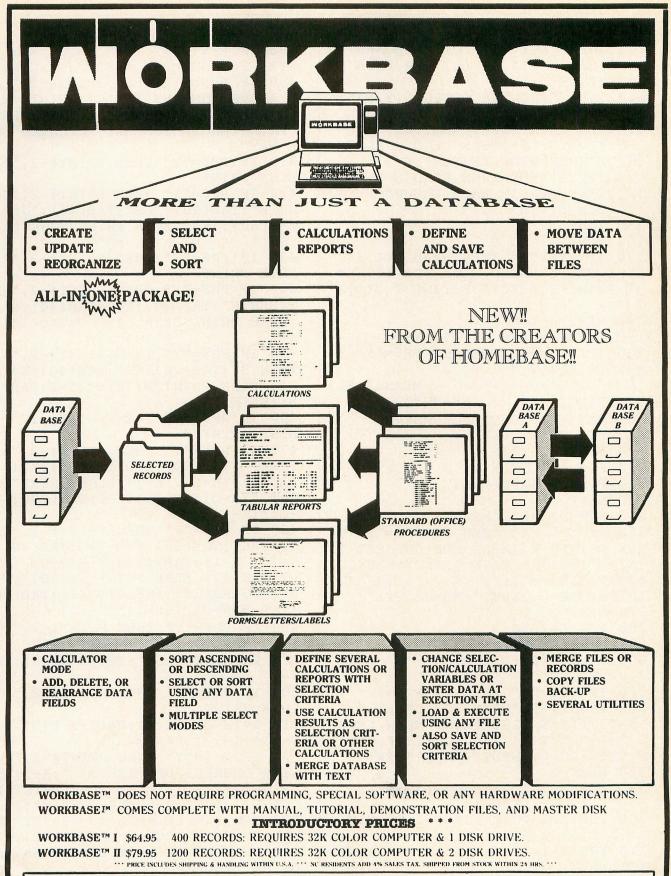
```
100 INPUT"KEY (I) FOR KEYBOARD I
NPUT OF  DATA OR (D) FOR DATA S
TATEMENTS.";YY$
110 INPUT"PRINTER OUTPUT (Y/N)";
PP$
115 REM DATA ENTRY ROUTINE.
120 IF YY$="D" THEN READ K ELSE
INPUT"NUMBER OF LEVELS";K
130 IF YY$="D" THEN READ N ELSE
INPUT"NUMBER IN EACH GROUP";N
140 DIM D(K,N)
150 FOR L=1 TO K
160 FOR S=1 TO N
170 IF YY$="D" THEN READ D(L,S)
ELSE PRINT"ENTER DATA FOR LEVEL"
;L:PRINT"SUBJECT";S;:INPUT D(L,S
)
180 NEXT S,L
190 INPUT"MEANS (Y/N)";YY$
200 IF YY$="Y" THEN IF PP$="Y" T
HEN PRINT#-2,"
                       MEANS" EL
SE PRINT"        MEANS"
205 REM FIND SUMS FOR MEANS.
210 DIM P(N),T(K)
220 FOR S=1 TO N
230 FOR L=1 TO K
240 P(S)=P(S)+D(L,S)
250 NEXT L
255 REM DISPLAY MEANS.
260 IF YY$="Y" THEN IF PP$="Y" T
HEN PRINT#-2,"MEAN FOR SUBJECT "
;S;"=";P(S)/K ELSE PRINT"MEAN SU
BJECT ";S;"=";P(S)/K:C=C+1:IF C>
10 THEN C=0:INPUT"(MORE)";XX
270 NEXT S
280 FOR L=1 TO K
290 FOR S=1 TO N
300 T(L)=T(L)+D(L,S)
310 NEXT S
320 IF YY$="Y" THEN IF PP$="Y" T
HEN PRINT#-2,"MEAN FOR LEVEL ";L
;"=";T(L)/N ELSE PRINT"MEAN LEVE
L ";L;"=";T(L)/N:CC=CC+1:IF CC>1
0 THEN CC=0:INPUT"(MORE)";XX
330 NEXT L
340 FOR X=1 TO N:G=G+P(X):NEXT
350 O=(G^2)/(K*N)
360 FOR L=1 TO K:FOR S=1 TO N:T=
T+(D(L,S)^2):NEXT S,L
370 FOR X=1 TO K:H=H+(T(X)^2):NE
XT:H=H/N
380 FOR X=1 TO N:F=F+(P(X)^2):NE
XT:F=F/K
385 REM DISPLAY SUMMARY TABLE
390 IF PP$<>"Y" THEN INPUT"KEY E
NTER FOR SUMMARY TABLE";XX:CLS
400 IF PP$="Y" THEN PRINT#-2,"
  ANOVA SUMMARY TABLE":PRINT#-2,
"REPEATED MEASURES ONE WAY" ELSE
PRINT"ANOVA SUMMARY REPEATED ON
E WAY"
410 IF PP$="Y" THEN PRINT#-2,""
ELSE PRINT
420 IF PP$="Y" THEN PRINT#-2,"SO
URCE";TAB(20);"SS";TAB(40);"df";
TAB(60);"MS" ELSE PRINT"SOURCE
  SS       DF       MS"
430 IF PP$="Y" THEN PRINT#-2,""
ELSE PRINT
440 IF PP$="Y" THEN PRINT#-2,"BE
TWEEN S";TAB(20);F-O;TAB(40);N-1
 ELSE PRINT"BETW. ";F-O;TAB(18);
N-1
450 IF PP$="Y" THEN PRINT#-2,"WI
THIN S";TAB(20);T-F;TAB(40);N*(K
-1) ELSE PRINT"WITHIN";T-F;TAB(1
8);N*(K-1)
460 IF PP$="Y" THEN PRINT#-2,""
ELSE PRINT""
470 IF PP$="Y" THEN PRINT#-2,"
TREAT";TAB(20);H-O;TAB(40);K-1;T
AB(60);(H-O)/(K-1) ELSE PRINT"TR
EAT ";H-O;TAB(18);K-1;TAB(20);(H
-O)/(K-1)
480 IF PP$="Y" THEN PRINT#-2,"RE
SIDUAL";TAB(20);T-H-F+O;TAB(40)(
N-1)*(K-1);TAB(60);(T-H-F+O)/((N
-1)*(K-1)) ELSE PRINT"RESID.";T-
H-F+O;TAB(18);(N-1)*(K-1);TAB(20
);(T-H-F+O)/((N-1)*(K-1))
490 IF PP$="Y" THEN PRINT#-2,"TO
TAL";TAB(20);T-O;TAB(40);K*N-1 E
LSE PRINT"TOTAL ";T-O;TAB(18);K*
N-1
500 IF PP$="Y" THEN PRINT#-2,""
ELSE PRINT
510 A=(H-O)/(K-1):B=(T-H-F+O)/((
N-1)*(K-1)):F=A/B
520 IF PP$="Y" THEN PRINT#-2,"F
WITH DEGREES OF FREEDOM ";K-1;",
";(N-1)*(K-1);"=";F ELSE PRINT"F
 (";K-1;",";(N-1)*(K-1);")=";F
522 REM SAMPLE DATA TO VERIFY EN
TRY OF THE PROGRAM (SEE TEXT).
523 REM DELETE THIS BEFORE SAVIN
G THE PROGRAM.
5000 DATA 2,4
5010 DATA 1,2,3,4
5020 DATA 1,2,4,6
```

Analyze financial statements the way your broker would.

INVESTOR'S ANALYST

by Daniel C. Jamet

I NVESTOR'S ANALYST WILL help you analyze a company's assets: common stocks, preferred stocks, and bonds for investment value. Serious investment in corporate stocks or bonds is complex because of the quantity of information that must be analyzed. The financial section of the daily paper can give you a stock's transaction volume, or quote the day's prices, but this is not nearly enough. When you invest in a company's stock, you become a part owner; by pur-

chasing bonds, you become a creditor of the company.

Before using the program you'll need the financial statements of the company in which you're considering investing. The analysis should be based on the com-

pany's latest annual report (available to the public from the company or through brokerage firms).

Run the program and the menu gives you the options of analyzing common stocks, preferred stocks, or bonds.(See Photo 1. The program should be used to analyze convertible preferred stocks as common stock.) Make your selection; you are then asked to enter data. You can locate all the data needed in the financial statements. Enter the data as

prompted; the ratios will be computed and displayed on screen, and the significance of each ratio will be given (as shown in Photo 1b).

Pointers

Here are a few pointers for potential investors. For common stocks investment, look first at each company's earnings record. Earnings per share and dividends per share are of particular interest. Potential preferred stockholders should be interested in the yield on their investment. The measurement of the safety of an investment in preferred stock is the firm's ability to meet its preferred dividend requirements. Potential bondholders should pay particular attention to these three factors:

- The rate of return on their investment;
- The firm's ability to meet its interest requirements;
- The firm's ability to repay the principal of the debt when it falls due.

Don't expect to buy stocks, watch their value increase, sell them, buy some more, and keep repeating this pleasant experience. It just doesn't happen that way. While it is necessary to identify the underlying trend over a long time period, it is impossible to spot the high and low points in the shorter swings. Even professionals can't do it consistently.

Possible goals for the careful investor might be: long-term appreciation with little emphasis on current yield; relatively high current income that will increase over the years; or a combination of both. It is also safer to invest with companies that have top-quality management and a strong financial position. These companies are among the innovative and aggressive leaders in a growing industry. Their sales, earnings and dividends are increasing and have good prospects of increasing in the future. ■ ■ ■

```
INPUT THE FOLLOWING INFORMATION

NET INCOME AVAILABLE FOR COMMON
STOCK=? 17467000

NUMBER SHARES OF COMMON STOCK
OUTSTANDING=? 254670

DIVIDENT PER SHARE=? 1.26

MARKET PRICE PER SHARE=? 36.3

EARNINGS PER SHARE=? 24.0

COMMON STOCKHOLDERS'
EQUITY=? 125930
```

Photo 1a. Data Entry Screen

```
EARNINGS PER SHARE ON COMMON
STOCK= $68.59

        SIGNIFICANCE

GIVES THE AMOUNT OF EARNINGS
APPLICABLE TO A SHARE OF
COMMON STOCK.



PRESS <ENTER> TO CONTINUE?
```

Photo 1b. Same Results Screen

Program Listing. Investor's Analyst

TAPE LOADER

16K Extended Color Basic

```
15 CLS
20 PRINT@6,"-INVESTOR'S ANALYST-
"
25 PRINT
30 PRINT"TO ANALYSE A COMPANY FO
R INVES-"
35 PRINT"TMENT PURPOSES ITS LATE
ST FINAN-"
40 PRINT"CIAL REPORT IS REQUIRED
."
45 PRINT:PRINT"(1)ANALYSIS FOR C
OMMON STOCK"
50 PRINT"   INVESTMENT"
55 PRINT"(2)ANALYSIS FOR PREFERR
ED"
60 PRINT"   STOCK INVESTMENT"
65 PRINT"(3)ANALYSIS FOR BOND"
70 PRINT"   INVESTMENT"
75 PRINT"(4)QUIT"
80 PRINT:PRINT"ENTER DESIRED ANA
LYSIS NUMBER"
85 Z$=INKEY$
90 IF Z$="" THEN 85
95 ON VAL(Z$) GOTO 100,550,805,1
050
100 CLS :'COMMON STOCK
105 PRINT"INPUT THE FOLLOWING IN
FORMATION"
110 PRINT
115 PRINT"NET INCOME AVAILABLE F
OR COMMON"
```

```
120 PRINT"STOCK=";
125 INPUT A
130 IF A=0 THEN 115
135 PRINT
140 PRINT"NUMBER OF SHARES OF CO
MMON STOCK"
145 PRINT"OUTSTANDING=";
150 INPUT B
155 IF B=0 THEN140
160 PRINT
165 PRINT"DIVIDENDS PER SHARE=";
170 INPUT C
175 IF C=0 THEN 165
180 PRINT
185 PRINT"MARKET PRICE PER SHARE
=";
190 INPUT D
195 IF D=0 THEN 185
200 PRINT
205 PRINT"EARNINGS PER SHARE=";
210 INPUT E
215 IF E=0 THEN 205
220 PRINT
225 PRINT"COMMON STOCKHOLDERS'"
230 PRINT"EQUITY=";
235 INPUT F
240 IF F=0 THEN 225
245 PRINT
250 PRINT"OPERATING EXPENSES=";
255 INPUT G
260 IF G=0 THEN 250
```

```
265 PRINT
270 PRINT"NET SALES=";
275 INPUT H
280 IF H=0 THEN 270
285 PRINT:PRINT"DO YOU NEED TO C
HANGE ANY OF"
290 PRINT"THE ABOVE VALUES? YES
OR NO";:INPUT P$
295 IF P$="YES" THEN 100
300 IF P$="Y" THEN 100
305 IF P$="N" THEN 310
310 LET EPS=A/B
315 LET DY=C/D
320 LET PER=D/E
325 LET BVS=F/B
330 LET OER=G/H
335 CLS
340 PRINT:PRINT"EARNINGS PER SHA
RE ON COMMON"
345 PRINT USING"STOCK=$$###.##";
EPS
350 PRINT
355 PRINT@138,"SIGNIFICANCE"
360 PRINT:PRINT"GIVES THE AMOUNT
 OF EARNINGS"
365 PRINT"APPLICABLE TO A SHARE
OF"
370 PRINT"COMMON STOCK."
375 PRINT@480,"PRESS <ENTER> TO
CONTINUE";:INPUT Z$
380 CLS
385 PRINT:PRINT USING"DIVIDEND Y
IELD=%###.##";DY
390 PRINT@138,"SIGNIFICANCE"
395 PRINT:PRINT"SHOWS THE RATE E
ARNED BY"
400 PRINT"STOCKHOLDERS BASED ON
CURRENT
405 PRINT"PRICE FOR A SHARE OF S
TOCK."
410 PRINT@480,"PRESS <ENTER> TO
CONTINUE";:INPUT Z$
415 CLS
420 PRINT:PRINT USING"PRICE-EARN
ING RATIO=##.#";PER
425 PRINT@138,"SIGNIFICANCE"
430 PRINT:PRINT"INDICATES WHETHE
R PRICE OF"
435 PRINT"STOCK IS IN LINE WITH
EARNINGS."
440 PRINT@480,"PRESS <ENTER> TO
CONTINUE";:INPUT Z$
445 CLS
```

```
450 PRINT:PRINT"BOOK VALUE PER S
HARE OF"
455 PRINT USING"COMMON STOCK=$$#
##.##";BVS
460 PRINT@138,"SIGNIFICANCE"
465 PRINT:PRINT"MEASURES THE REC
ORDED VALUE"
470 PRINT"OF NET ASSETS BEHIND E
ACH"
475 PRINT"SHARE OF STOCK."
480 PRINT@480,"PRESS <ENTER> TO
CONTINUE";:INPUT Z$
485 CLS
490 PRINT:PRINT USING"OPERATING
EXPENSE RATIO=##.#";OER
495 PRINT@138,"SIGNIFICANCE"
500 PRINT:PRINT"INDICATES MANAGE
MENT'S ABILITY"
505 PRINT"TO CONTROL EXPENSES."
510 PRINT:PRINT"WOULD YOU LIKE T
O DISPLAY"
515 PRINT"RATIOS AGAIN? YES OR N
O";:INPUT P$
520 IF P$="YES" THEN 335
525 IF P$="Y" THEN 335
530 IF P$="NO" THEN 540
535 IF P$="N" THEN 540

540 PRINT@480,"PRESS <ENTER> FOR
MENU";:INPUT Z$
545 GOTO 15
550 CLS:'PREFERRED STOCK
555 PRINT"INPUT THE FOLLOWING IN
FORMATION"
560 PRINT
565 PRINT"DIVIDEND PER SHARE OF
PREFERRED"
570 PRINT"STOCK=";
575 INPUT A
580 IF A=0 THEN 565
585 PRINT
590 PRINT"MARKET PRICE PER SHARE
 OF"
595 PRINT"PREFERRED STOCK=";
600 INPUT B
605 IF B=0 THEN 590
610 PRINT
615 PRINT"NET INCOME AVAILABLE F
OR"
620 PRINT"PREFERRED STOCK=";
625 INPUT C
630 IF C=0 THEN 615
635 PRINT
640 PRINT"ANNUAL PREFERRED DIVID
ENDS=";
```

```
645 INPUT D
650 IF D=0 THEN 640
655 PRINT:PRINT"DO YOU NEED TO C
HANGE ANY OF"
660 PRINT"THE ABOVE VALUES? YES
OR NO";:INPUT P$
665 IF P$="YES" THEN 550
670 IF P$="Y" THEN 550
675 IF P$="NO" THEN 685
680 IF P$="N" THEN 685
685 LET PDY=A/B
690 LET TDE=C/D
695 CLS
700 PRINT:PRINT USING"DIVIDEND Y
IELD=%###.##";PDY
705 PRINT@138,"SIGNIFICANCE"
710 PRINT:PRINT"SHOWS THE RATE E
ARNED BY"
715 PRINT"STOCKHOLDERS BASED ON
CURRENT"
720 PRINT"PRICE FOR A SHARE OF S
TOCK."
725 PRINT@480,"PRESS <ENTER> TO
CONTINUE";:INPUT Z$
730 CLS
735 PRINT:PRINT"TIMES PREFERRED
DIVIDENDS"
740 PRINT USING"EARNED=##.#";TDE
745 PRINT@138,"SIGNIFICANCE"
750 PRINT:PRINT"SHOWS THE ADEQUA
CY OF CURRENT"
755 PRINT"EARNINGS TO PAY DIVIDE
NDS ON"
760 PRINT"PREFERRED STOCK."
765 PRINT:PRINT"WOULD YOU LIKE T
O DISPLAY"
770 PRINT"RATIOS AGAIN? YES OR N
O";:INPUT P$
775 IF P$="YES" THEN 695
780 IF P$="Y" THEN 695
785 IF P$="NO" THEN 795
790 IF P$="N" THEN 795
795 PRINT@480,"PRESS <ENTER> FOR
 MENU";:INPUT Z$
800 GOTO 15
805 CLS:'BOND
810 PRINT"INPUT THE FOLLOWING IN
FORMATION"
815 PRINT
820 PRINT"OPERATING INCOME=";
825 INPUT A
830 IF A=0 THEN 820
835 PRINT
840 PRINT"ANNUAL INTEREST EXPENS
E=";
845 INPUT B
```

```
850 IF B=0 THEN 840
855 PRINT
860 PRINT"TOTAL LIABILITIES=";
865 INPUT C
870 IF C=0 THEN 860
875 PRINT
880 PRINT"TOTAL ASSETS=";
885 INPUT D
890 IF D=0 THEN 880
895 PRINT:PRINT"DO YOU NEED TO C
HANGE ANY OF"
900 PRINT"THE ABOVE VALUES? YES
OR NO";:INPUT P$
905 IF P$="YES" THEN 805
910 IF P$="Y" THEN 805
915 IF P$="NO" THEN 925
920 IF P$="N" THEN 925
925 LET NIE=A/B
930 LET DR=C/D
935 CLS
940 PRINT:PRINT"NUMBER OF TIMES
INTEREST"
945 PRINT USING"EARNED=##.#";NIE
950 PRINT@138,"SIGNIFICANCE"
955 PRINT:PRINT"MEASURES THE COV
ERAGE OF INTER-"
960 PRINT"EST REQUIREMENTS, PART
ICULARLY"
965 PRINT"ON LONG TERM DEBT."
970 PRINT@480,"PRESS <ENTER> TO
CONTINUE";:INPUT Z$
975 CLS
980 PRINT:PRINT USING"DEBT RATIO
=%###.##";DR
985 PRINT@138,"SIGNIFICANCE"
990 PRINT:PRINT"INDICATES THE PE
RCENTAGE OF"
995 PRINT"ASSETS FINANCED THROUG
H BOR-"
1000 PRINT"ROWING; IT SHOWS THE
EXTENT"
1005 PRINT"OF TRADING ON THE EQU
ITY."
1010 PRINT:PRINT"WOULD YOU LIKE
TO DISPLAY"
1015 PRINT"RATIOS AGAIN? YES OR
NO";:INPUT P$
1020 IF P$="YES" THEN 935
1025 IF P$="Y" THEN 935
1030 IF P$="NO" THEN 1040
1035 IF P$="N" THEN 1040
1040 PRINT@480,"PRESS <ENTER> FO
R MENU";:INPUT Z$
1045 GOTO 15
1050 END
```
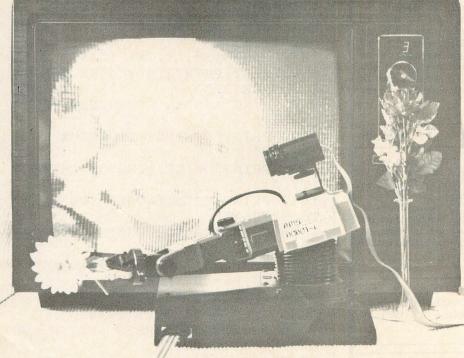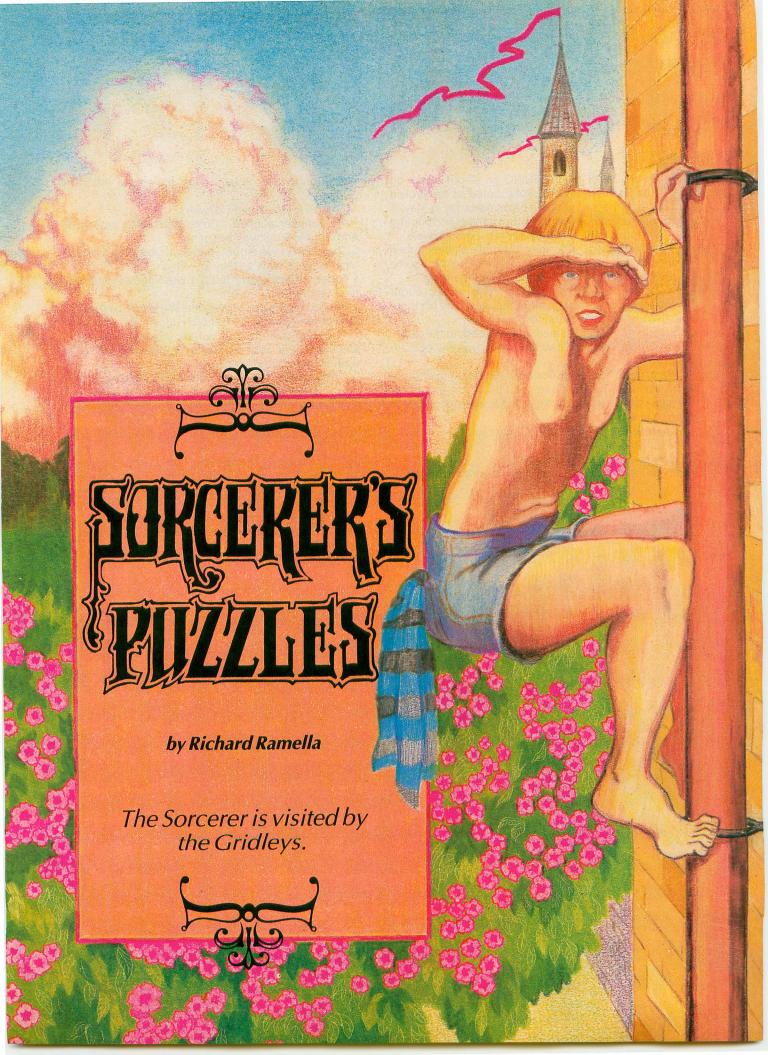
■■■

SORCERER'S PUZZLES

by Richard Ramella

The Sorcerer is visited by the Gridleys.

SPRING WAS THREATENING an all-out attack on the Sorcerer's castle, Poke noted as he strolled along the parapet on a balmy Saturday in May. A profusion of rainbow weed, prism pansies and bunting vine lapped happily at the base of the granite walls.

The boy did a little jig on the grave of winter, then decided to shinny down a rainspout to pick some wildflowers for Sunflower. It seemed only fitting.

Before he could hop over the crenel, his eye caught a wide, sweeping movement in the west. The hill there was aswarm with... Somethings!

"Invasion!" Poke screamed, running back to the castle. "Fill the moat! Man the battlements! Lower the portcullis!"

"Here, here!" Sorcerer Donald E. Goldberg said somewhat testily. He was trying to get a bit of sun and nap at the same time. He rose from a rather handsome carved patio chair and shaded his eyes.

"There are hundreds of them!" Poke said.

"Yes, there are, lad."

"How will we defend ourselves?"

"By giving them copious amounts of strawberry seltzer tonic. Don't worry, Poke, it's just the annual visit of the Gridleys."

"As in *Fire when ready, Gridleys?*"

The Sorcerer laughed. "Go tell Bob there's work to do. And you and Sunflower help him."

The Gridleys were good-humored, roly-poly creatures. Most were as wide as they were high. Short of ordering a battery of laboratory tests, it was impossible to tell if they were human.

It fell to Sunflower to welcome the leader of the thousand or so creatures — a General Pudge.

"Hut-two-three-four. Hut-two-*howdy!*-three-four!" said the general as he entered the room. He didn't simply walk. He marched at 90-degree angles.
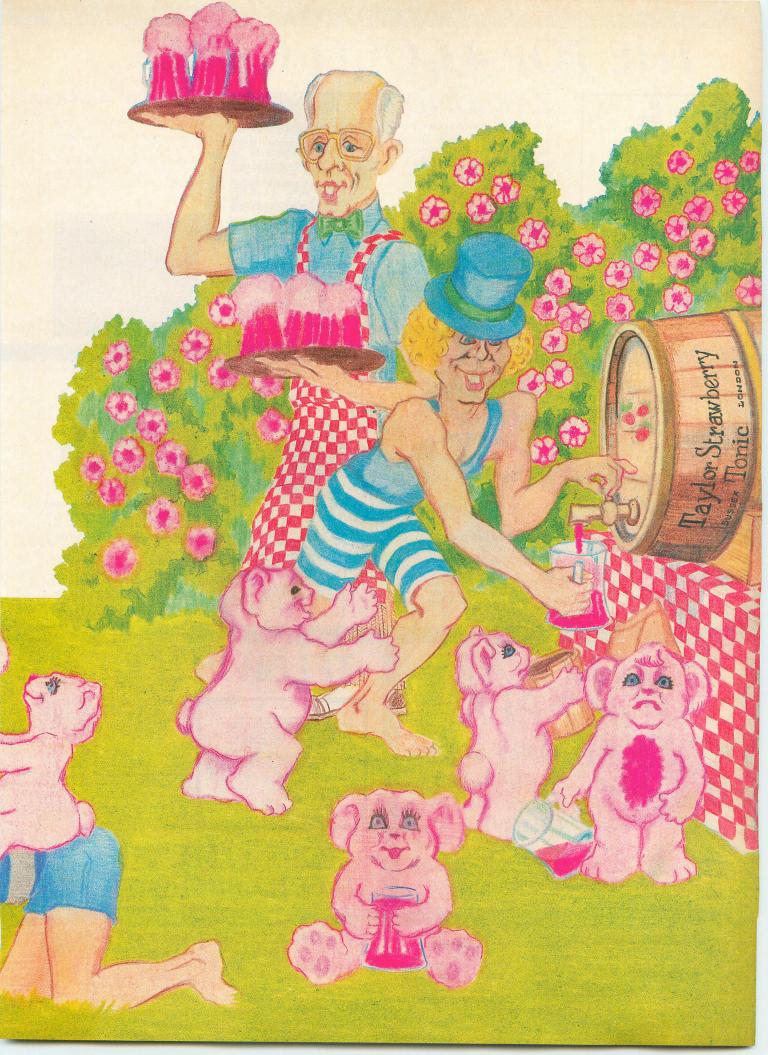
"Welcome to the Sorcerer's castle," the girl said.

"We're happy to be here," said General Pudge.

"I'll show you to your room," Sunflower offered.

"I know where it is, dumpling. Want to dance?"

"No, there's no music. And I'm not a dumpling."

"That's okay," said Pudge. "There's a song in my heart." And with that he hut-two-three-foured away. Sunflower followed, watching this curious, restrained style of dancing. Pudge did not abandon his 90-degree travel. This caused him to traverse the entire labyrinth of the Four Great Blue Hallways of the castle, until he came to his room, which was only a few steps from where he began his trek (see puzzle *"Square Dance"*).

"Say, I almost forgot!" Pudge said, snapping his fat little fingers with limited success. "I left my four ponies in the stable, and I need someone to go down there and switch their positions."

"All right," Sunflower said doubtfully, "what do I do?"

Pudge grinned. "Those silly things believe they're chess pieces, and that's the only way they'll move. The magenta ones and the orange ones have to be switched. You'll figure it out." (see *"Knightly Pursuits"*.)

Sunflower started to leave, then turned. "Why do all you Gridleys only go places in straight lines and right angles?"

Pudge started. "What do you know?" he asked.

"Nothing," said Sunflower. "I was just asking."

"Straight lines and right angles are the most efficient means of movement," the general said with a frown. "Right?"

"Not necessarily," the girl said.

"Ah-hah, you do know!" The little general moved very close. "You've heard of our research," he whispered.

"What research?"

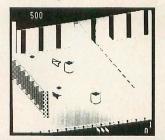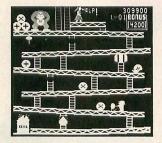"Observe," said Pudge. He pointed to a chair. "I want to go to that chair, but I am facing somewhat away from it. To reach the chair, I march five steps forward, do a sharp left face and approach the chair."

"If you're so inclined," Sunflower allowed.

"But then there's the Diagon Theory," Pudge said. "It's still experimental, but our scientists are moving closer every day to irrefutable proof that to reach that very chair, it is only necessary to walk diagonally toward it."

"I shall hope night and day for success," Sunflower smiled.

"You're a kind girl," Pudge said, "and for being so nice, I will show you the preliminary results of our research this evening after I've had a few strawberry seltzer tonics."

"You're going to attempt diagonal movement?"

"Do I look crazy?" Pudge asked. "No, we have a little test model for diagonal movement, and I shall let you have a go

at (and he whispered in the smallest of voices) The Diagons!"

Sorcerer's Notes

"Square Dance works in 4K Color Basic and 4K MC-10 Basic. Four overlapping blue rectangles are drawn, and a flashing orange cursor appears. The object is to turn the entire network of lines to orange by traveling it. First, take the orange cursor to the starting position you want by tapping the letter A for north, Z for south, the comma key for west, and the period key for east. When ready to challenge the pathway, tap the asterisk key, holding down the shift key as you do. At this point, the number 215 appears at top left screen. Now your travels will leave a trail of orange, and the number will be reduced by one each time a position of the network turns orange. If, the first time you arrive at any intersection, you leave it by turning left or right, the intersection will remain blue, allowing you to travel through it a second time. When the network is entirely orange, you will be recognized as a winner. You will have to recognize losing positions for yourself and start over.

"Knightly Pursuits" works in 16K Color Basic and 20K MC-10 Basic. A stable with nine stalls in a three-by-three grid holds four horses: magenta in the northwest and northeast stalls, orange at the southwest and southeast. Object is to reverse the positions, ending with orange ponies at northwest and northeast, magenta at southwest and southeast. A small flashing cursor may be moved among the stalls by tapping the arrow keys. To move a horse, position the cursor in the stall of the horse that will travel and tap M for move. The word WHERE? will appear on the screen and then be erased. The horses' moves must mimic those of a knight on a chess board — two squares in one direction and then

one left or right. To select a move, bear in mind W stands for west, N for north, E for east and S for south. Then tap any of the following eight applicable compass directions: NNW, WWN, WWS, SSW, SSE, EES, EEN, NNE. You will see your direction displayed at the top right of the screen. If nothing happens when you attempt to move, the move is illegal, such as attempted moves out of the stable, into an occupied stall, or from an unoccupied stall. To erase a botched move entry, tap X. The computer recognizes a winning position. It also knows the minimum number of moves required to win and will inform you if you win in that number. Otherwise, it will not reveal the number but will let you know you can better your score.

"The Diagons" works only in 16K Extended Basic. A four-by-five playing board is drawn. In the top row are magenta diagons, and the bottom row has orange diagons. The object is to reverse the colors by moving according to strict rules. Note a small dark circle over the diagon in the northwest corner square. This is a cursor and may be moved by tapping the arrow keys. This cursor must be positioned on the diagon to move. To make a move, two choices must be made in turn. First, tap 1, 2 or 3 for the number of squares you will move the piece diagonally. Next, tap any of the following letters to take the piece in the direction they represent: A — northwest, S — northeast, Z — southwest, X — southeast. Both colors of diagons may move up and down. The game is lost if two diagons of different colors wind up in any diagonal row. A winning position is recognized by the computer. There is a rare situation when an incorrect move command can put diagons in the same slanting row without recognizing the losing pattern until the next move has been made. Consider this a secret present from the Sorcerer. ∎ ∎ ∎

4K Color Basic

Program Listing. Square Dance

```
120 CLS
130 CLEAR 100
140 PRINT @ 233,"SQUARE DANCE";
150 FOR T=1 TO 30
160 SOUND RND(8)*13,1
170 NEXT
180 U$="A"
190 D$="Z"
200 L$=","
210 R$="."
220 Y$="WINNER"
```

```
230 M=179
240 PK=1024
250 REM * For MC-10 change above
 line to PK=16384
260 CLS0
270 B$=CHR$(175)
280 O$=CHR$(255)
290 FOR A=1 TO 19
300 A$=A$+B$
310 NEXT
320 FOR X=44 TO 230 STEP 62
330 PRINT @ X,A$;
340 PRINT @ X+256,A$;
350 FOR Z=X TO X+256 STEP 32
360 PRINT @ Z,B$;
370 PRINT @ Z+19,B$;
380 NEXT Z,X
390 PRINT @ M,O$;
400 X$=INKEY$
410 IF X$="*" THEN H=215: GOTO 4
70
420 S=M
430 GOSUB 600
440 PRINT @ S,B$;
450 PRINT @ M,O$;
460 GOTO 400
470 X$=INKEY$
480 S=M
490 GOSUB 600
500 PRINT @ M,O$;
510 PRINT @ 0,H;
520 IF H=0 THEN 690
530 IF S=108 OR S=170 OR S=172 O
R S=315 OR S=317 OR S=379 THEN P
RINT @ S,B$;
540 IF S=236 OR S=234 OR S=232 O
R S=313 OR S=377 OR S=441 THEN P
RINT @ S,B$;
550 IF X$=D$ AND PEEK(PK+M-64)=2
55 THEN PRINT @ M-32,O$;
560 IF X$=U$ AND PEEK(PK+M+64)=2
55 THEN PRINT @ M+32,O$;
570 IF X$=L$ AND PEEK(PK+M+2)=25
5 THEN PRINT @ M+1,O$;
580 IF X$=R$ AND PEEK(PK+M-2)=25
5 THEN PRINT @ M-1,O$;
590 GOTO 470
600 U=PEEK(PK+M-32)
610 D=PEEK(PK+M+32)
620 L=PEEK(PK+M-1)
630 R=PEEK(PK+M+1)
640 IF X$=U$ AND U=175 THEN M=M-
32: H=H-1
650 IF X$=D$ AND D=175 THEN M=M+
32: H=H-1
660 IF X$=L$ AND L=175 THEN M=M-
1: H=H-1
```

```
670 IF X$=R$ AND R=175 THEN M=M+
1: H=H-1
680 RETURN
690 J=1
700 FOR C=131 TO 291 STEP 32
710 PRINT @ C,MID$(Y$,J,1);
720 J=J+1
730 NEXT
740 SOUND 100+RND(10)*10,1
750 GOTO 740
760 END
```

■■■

Program Listing.
Knightly Pursuits

16K Color Basic

```
110 REM * SORCERER'S PUZZLES #7
* RICHARD RAMELLA *
120 CLEAR 500
130 CLS0
140 Z$="WHERE?"
150 U$=CHR$(94)
160 D$=CHR$(10)
170 L$=CHR$(8)
180 R$=CHR$(9)
190 PRINT @ 232,"KNIGHTLY PURSUI
TS";
200 K=0
210 Z=1
220 FOR L=50 TO 0 STEP -15
230 N=N+1
240 IF N/2=INT(N/2) THEN V=8: GO
TO 260
250 V=7
260 GOSUB 660
270 NEXT L
280 FOR T=1 TO 1500
290 NEXT T
300 CLS0
310 FOR Y=0 TO 30 STEP 10
320 FOR X=0 TO 60
330 SET(X,Y,5)
340 NEXT X,Y
350 FOR X=0 TO 60 STEP 20
360 FOR Y=1 TO 29
370 SET(X,Y,5)
380 NEXT Y,X
390 V=7
400 K=2
410 L=6
420 GOSUB 660
430 L=46
440 GOSUB 660
450 V=8
```

```
460 K=22
470 L=6
480 GOSUB 660
490 L=46
500 GOSUB 660
510 A=2
520 B=2
530 SET(A,B,3)
540 A$=INKEY$
550 IF POINT(9,2)=8 AND POINT(49
,2)=8 AND POINT(9,22)=7 AND POIN
T(49,22)=7 THEN 1910
560 IF A$="M" AND POINT(A+7,B)<>
0 THEN 790
570 G=A
580 H=B
590 IF A$=U$ AND B>2 THENB=B-10
600 IF A$=D$ AND B<22 THEN B=B+1
0
610 IF A$=L$ AND A>6 THENA=A-20
620 IF A$=R$ AND A<42 THEN A=A+2
0
630 RESET(G,H)
640 SET(A,B,2)
650 GOTO 540
660 FOR Y=K TO K+9
670 Z=Z+1
680 IF Z=8 THEN FOR X=L+2 TO L+6
: SET(X,Y,V): NEXT X
690 IF Z=2 THEN FOR X=L+2 TO L+7
: SET(X,Y,V): NEXT X
700 IF Z=3 THEN FOR X=L TO L+11:
 SET(X,Y,V): NEXT X
710 IF Z=4 THEN FOR X=L+2 TO L+1
0: SET(X,Y,V): NEXT X
720 IF Z=5 THEN FOR X=L TO L+6:
SET(X,Y,V): NEXT X
730 IF Z=6 THEN FOR X=L+2 TO L+6
: SET(X,Y,V): NEXT X
740 IF Z=7 THEN FOR X=L TO L+6:
SET(X,Y,V): NEXT X
750 IF Z=9 THEN FOR X=L TO L+9:
SET(X,Y,V): NEXT X
760 NEXT Y
770 Z=1
780 RETURN
790 P=1
800 FOR J=31 TO 191 STEP 32
810 PRINT @ J,MID$(Z$,P,1);
820 SOUND 40+J,1
830 P=P+1
840 NEXT J
850 FOR T=1 TO 300
860 NEXT T
870 FOR J=31 TO 191 STEP 32
880 PRINT @ J,CHR$(128);
890 SOUND 40+J,1
900 NEXT J
910 P=1
920 A$=""
930 A$=A$+INKEY$
940 IF LEN(A$)>3 OR RIGHT$(A$,1)
="X" THEN GOSUB 1790
950 FOR J=31 TO LEN(A$)*32 STEP
32
960 PRINT @ J,MID$(A$,P,1);
970 P=P+1
980 NEXT J
990 IF A$<>"NNW" THEN 1060
1000 IF A=2 OR B<22 GOSUB 1790:
GOTO 540
1010 IF POINT(A-13,B-20)<>0 THEN
 GOSUB 1790: GOTO 540
1020 Q=A
1030 W=B
1040 GOSUB 1840
1050 K=B-20: L=A-17: GOSUB 660:
GOTO 540
1060 IF A$<>"WWN" THEN 1160
1070 IF A<42 OR B=2 GOSUB 1790:
GOTO 540
1080 IF POINT(9,B-10)<>0 GOSUB 1
790: GOTO 540
1090 Q=A
1100 W=B
1110 GOSUB 1840
1120 K=B-10
1130 L=A-36
1140 GOSUB 660
1150 GOTO 540
1160 IF A$<>"WWS" THEN 1260
1170 IF A<42 OR B=22 GOSUB 1790:
 GOTO 540
1180 IF POINT(A-33,B+10)<>0 GOSU
B 1790: GOTO 540
1190 Q=A
1200 W=B
1210 GOSUB 1840
1220 K=B+10
1230 L=A-36
1240 GOSUB 660
1250 GOTO540
1260 IF A$<>"SSW" THEN 1360
1270 IF A=2 OR B>2 GOSUB 1790: G
OTO 540
1280 IF POINT(A-13,B+20)<>0 GOSU
B 1790: GOTO 540
1290 Q=A
1300 W=B
1310 GOSUB 1840
1320 K=B+20
1330 L=A-16
```

```
1340 GOSUB660
1350 GOTO 540
1360 IF A$<>"SSE" THEN 1460
1370 IF A=42 OR B>2 GOSUB 1790:
GOTO 540
1380 IF POINT(A+27,B+20)<>0 GOSU
B 1790: GOTO 540
1390 Q=A
1400 W=B
1410 GOSUB 1840
1420 K=B+20
1430 L=A+24
1440 GOSUB 660
1450 GOTO 540
1460 IF A$<>"EES" THEN 1560
1470 IF A>2 OR B=22 GOSUB 1790:
GOTO 1890
1480 IF POINT(A+49,B+10)<>0 GOSU
B 1790: GOTO 540
1490 Q=A
1500 W=B
1510 GOSUB 1840
1520 K=B+10
1530 L=A+43
1540 GOSUB 660
1550 GOTO 540
1560 IF A$<>"EEN" THEN 1660
1570 IF A>2 OR B=2 GOSUB 1790: G
OTO 540
1580 IF POINT(A+47,B-10)<>0 GOSU
B 1790: GOTO 540
1590 Q=A
1600 W=B
1610 GOSUB 1840
1620 K=B-10
1630 L=A+43
1640 GOSUB 660
1650 GOTO 540
1660 IF A$<>"NNE" THEN 1770
1670 IF A=42 OR B<22 GOSUB 1790:
 GOTO 540
1680 IF POINT(A+27,B-20)<>0 GOSU
B 1790: GOTO 540
1690 Q=A
1700 W=B
1710 GOSUB 660: GOSUB 1840
1720 K=B-20
1730 L=A+23
1740 GOSUB 660
1750 GOSUB 1790
1760 GOTO 540
1770 P=1
1780 GOTO 930
1790 A$=""
1800 FOR J=31 TO 191 STEP 32
1810 PRINT @ J,CHR$(128);
1820 NEXT J
1830 RETURN
1840 V=POINT(A+7,B)
1850 TY=TY+1
1860 FOR Y=B TO B+7
1870 FOR N=Q TO Q+15
1880 RESET(N,Y)
1890 NEXT N,Y
1900 RETURN
1910 PRINT @ 225,"WINNER IN"TY"M
OVES.";
1920 IF TY=1+(ASC(U$)-14)/5 THEN
 PRINT @ 257,"LOWEST POSSIBLE SC
ORE": GOTO 1920
1930 PRINT @ 257,"YOU CAN DO BET
TER, THOUGH...";: GOTO 1930
1940 END                    ■■■
```

Program Listing. The Diagons

```
110 REM * SORCERER'S PUZZLES #7
* RICHARD RAMELLA *
120 CLS
130 CLEAR 100
140 PRINT @ 234,"THE DIAGONS";
150 FOR T=1 TO 30
160 SOUND RND(8)*13,1
170 NEXT
180 CLS0
190 PCLEAR 4
200 PMODE 3,1
210 PCLS1
220 DIM C(10,10),D(10,10),E(10,1
0),F(20)
230 U$=CHR$(94)
240 D$=CHR$(10)
250 L$=CHR$(8)
260 R$=CHR$(9)
270 GET(10,7)-(23,23),C
280 CIRCLE(16,16),9,3
290 PAINT(16,16),3,3
300 GET(10,7)-(26,25),D
310 PUT(8,7)-(26,25),C
320 CIRCLE(16,16),9,4
330 PAINT(16,16),4,4
340 GET(10,7)-(26,25),E
350 PUT(8,7)-(26,25),C
360 SCREEN1,1
370 FOR X=0 TO 128 STEP 32
380 LINE(0,X)-(128,X),PSET
390 LINE(X,0)-(X,160),PSET
400 NEXT X
410 LINE(0,X)-(128,X),PSET
420 Y=7
```

```
430 FOR X=8 TO 104 STEP 32
440 SOUND X+Y,1
450 PUT(X,Y)-(X+19,Y+18),D
460 SOUND X+Y,1
470 PUT(X,Y+128)-(X+19,Y+146),E
480 NEXT
490 X=16
500 Y=16
510 CIRCLE(X,Y),4,2
520 A$=INKEY$
530 IF A$="" THEN 520
540 IF A$>"0" AND A$<"4" THEN L=
VAL(A$): GOTO 520
550 V=X
560 W=Y
570 IF A$=U$ AND Y>16 THEN Y=Y-3
2
580 IF A$=D$ AND Y<144 THEN Y=Y+
32
590 IF A$=L$ AND X>16 THEN X=X-3
2
600 IF A$=R$ AND X<112 THEN X=X+
32
610 CIRCLE(V,W),4,PPOINT(V,W)
620 CIRCLE(X,Y),4,2
630 G=PPOINT(X,Y)
640 S=PPOINT(X,Y)
650 IF S<>7 AND S<>8 THEN 520
660 IF L=0 THEN 520
670 L=L-1
680 IF X=16 OR Y=16 THEN 690 ELS
E IF PPOINT(X-32,Y-32)<>5 THEN 6
90 ELSE IF A$="A" THEN V=X-36: W
=Y-41: GOSUB 740
690 IF X=112 OR Y=16 THEN 700 EL
SE IF PPOINT(X+32,Y-32)<>5 THEN
700 ELSE IF A$="S" THEN V=X+28:
W=Y-41: GOSUB 740
700 IF X=16 OR Y=144 THEN 710 EL
SE IF PPOINT(X-32,Y+32)<>5 THEN
710 ELSE IF A$="Z" THEN V=X-36:
W=Y+23: GOSUB 740
710 IF X=112 OR Y=144 THEN 720 E
LSE IF PPOINT(X+32,Y+32)<>5 THEN
 720 ELSE IF A$="X" THEN V=X+28:
 W=Y+23: GOSUB 740
720 IF L>0 THEN 620
730 GOTO 520
740 PUT(X-6,Y-8)-(X+13,Y+9),C
750 IF G=8 THEN PUT(V,W)-(V+18,W
+18),E ELSE PUT(V,W)-(V+18,W+18)
,D
760 SOUND X+Y-10,1
770 IF A$="A" THEN X=X-32: Y=Y-3
2
780 IF A$="S" THEN X=X+32: Y=Y-3
2
```

The HJL-57 Keyboard

Compare it with the rest. Then, buy the best.

If you've been thinking about spending good money on a new keyboard for your Color Computer, why not get a good keyboard for your money?

Designed from scratch, the HJL-57 Professional Keyboard is built to unlock ALL the potential performance of your Color Computer. Now, you can do real word processing and sail through lengthy listings...with maximum speed; minimum errors.

At $79.95, the HJL-57 is reasonably priced, but you can find other CoCo keyboards for a few dollars less. So, before you buy, we suggest that you compare.

Compare Design.

The ergonomically-superior HJL-57 has sculptured, low profile keycaps; and the three-color layout is identical to the original CoCo keyboard.

Compare Construction.

The HJL-57 has a rigidized aluminum baseplate for solid, no-flex mounting. Switch contacts are rated for 100 million cycles minimum, and covered by a spill-proof membrane.

Compare Performance.

Offering more than full-travel, bounce-proof keyswitches, the HJL-57 has RFI/EMI shielding that eliminates irritating noise on displays; and four user-definable function keys (one latchable), specially-positioned to avoid inadvertent actuation.

Free Function Key Program

Your HJL-57 kit includes usage instructions and decimal codes produced by the function keys, plus a free sample program that defines the function keys as follows: F1 = Screen dump to printer. F2 = Repeat key (latching). F3 = Lower case upper case flip (if you have lower case capability). F4 = Control key; subtracts 64 from the ASCII value of any key pressed. Runs on disc or tape; extended or standard Basic.

Compare Installation.

Carefully engineered for easy installation, the HJL-57 requires no soldering, drilling or gluing. Simply plug it in and drop it right on the original CoCo

mounting posts. Kit includes a new bezel for a totally finished conversion.

Compare Warranties.

The HJL-57 is built so well, it carries a full, one-year warranty. And, it is sold with an exclusive 15-day money-back guarantee.

Compare Value.

You know that a bargain is a bargain only so long as it lasts. If you shop carefully, we think you will agree...The HJL-57 is the last keyboard your CoCo will ever need. And that's **real** value.

Order Today.

Only $79.95, the HJL-57 is available for immediate shipment for either the original Color Computer (sold prior to October, 1982) or the F-version and TDP-100 (introduced in October, 1982), and the new 64K CoCo. **Now also available for CoCo 2.**

Order by Phone Anytime

716-235-8358

24 hours, 7 days a week

HJL PRODUCTS INC.

955 Buffalo Road • P.O. Box 24954
Rochester, New York 14624

Ordering Information: Specify model (Original, F-version, or CoCo 2). Payment by C.O.D., check, MasterCard or Visa. Credit card customers include complete card number and expiration date. Add $2.00 for shipping ($3.50 for Canada). New York state residents add 7% sales tax. Dealer inquiries invited. For dealer information in Eastern U.S. and Canada, call collect: 617-586-7614, Advanced Computer Services (distributor), 74 Plain Street, Brockton, MA 02401.

Circle No. 33 on Reader Service Card

```
790 IF A$="Z" THEN X=X-32: Y=Y+3
2
800 IF A$="X" THEN X=X+32: Y=Y+3
2
810 CIRCLE(X,Y),4,2
820 IF L>0 THEN RETURN
830 J=1
840 FOR Q=16 TO 144 STEP 32
850 FOR R=16 TO 112 STEP 32
860 U=PPOINT(R,Q)
870 IF U=7 THEN F(J)=1 ELSE IF U
=8 THEN F(J)=2 ELSE F(J)=0
880 J=J+1
890 NEXT R,Q
900 IF F(1)+F(2)+F(3)+F(4)=8 AND
 F(17)+F(18)+F(19)+F(20)=4 THEN
1230
910 K=F(3)+F(8)
920 GOSUB 1160
930 K=F(2)+F(7)+F(12)
940 GOSUB 1160
950 K=F(1)+F(6)+F(11)+F(16)
960 GOSUB 1160
970 K=F(5)+F(10)+F(15)+F(20)
980 GOSUB 1160
990 K=F(9)+F(14)+F(19)
1000 GOSUB 1160
1010 K=F(13)+F(18)
1020 GOSUB 1160
1030 K=F(2)+F(5)
1040 GOSUB 1160
1050 K=F(3)+F(6)+F(9)
1060 GOSUB 1160
1070 K=F(4)+F(7)+F(10)+F(13)
1080 GOSUB 1160
1090 K=F(8)+F(11)+F(14)+F(17)
1100 GOSUB 1160
1110 K=F(12)+F(15)+F(18)
1120 GOSUB 1160
1130 K=F(16)+F(19)
1140 GOSUB 1160
1150 RETURN
1160 IF K>2 AND K/2<>INT(K/2) TH
EN 1170 ELSE RETURN
1170 PRINT@RND(63)*8,"LOSER !";
1180 SOUND 5,10
1190 SCREEN1,1
1200 SOUND 45,5
1210 SOUND 89,5
1220 GOTO 1170
1230 PRINT@RND(63)*8,"WINNER!";
1240 SOUND RND(100)+125,5
1250 SCREEN 1,1
1260 SOUND RND(100)+100,8
1270 GOTO 1230
1280 END
```

FLY

16K
Color Basic

EDTASM+

Rod McCormick

FLY, BY RONALD BOULANGER, is the first runner up in the assembly language category of 1983's Summer Programming Project. "Fly" is fast, and will soon have your head buzzing. Here's the author's description, and directions:

The object of the game is to catch as many flies as you can in three minutes. Each fly caught creates an obstacle. Be careful, you may be trapped by those obstacles. The more flies you catch, the tougher the game becomes. Each fly caught gives you one point.

Type CSAVE and Run the Basic loader. Entering Y to the ARE YOU READY? prompt will run the game. Pressing Break will re-

turn you to Basic so you can prepare a tape; type CSAVEM "FLY", 8192,8895,8192 to save the shorter machine code.

On top of the screen, the score board is shown. From left to right is the actual score (sc); the high score (hs); and the time (three minutes to begin). The bottom of the screen is the prompt: SELECT SPEED (1 – 9). The lower the number you press, the faster the game will be.

At this point, only keys 1 to 9 and the Break key are operating. The Break key will initiate a new game.

The game is played with the cursor control keys (arrow keys) up, down, left, right. Once the speed has been selected, the game starts. The blue dot is the fly,

controlled by the computer. The white dot is controlled by the player using the arrow keys. To score points, touch the blue dot with a white dot. Once you touch the fly it will freeze and you will hear a tone. This will create an obstacle and give you one point.

If you or the fly are trapped between obstacles, depress letter T for trap; this will stop the game and check for a new high score.

To play again, just select a speed. You can change speed any time in the game by depressing the letter S (for select). This will freeze the time left; once your speed selection has been made, the game continues where it stopped. ■ ■ ■

Program Listing. Basic Loader

16K Color Basic

```
0 'Loader for FLY, R. Boulanger
1 CLS: CLEAR200,8191: PRINT@204,
"FLY LOADER": PRINT@384
2 FOR AD=8192 TO 8895 STEP 32
3 LN=LN+10: CK=0: FOR M=AD TO AD
+31: READ D: CK=CK+D: POKE M,D:
NEXT M
4 READ CH: IF CH<>CK THEN PRINT"
 DATA TYPO IN LINE"LN: PRINT" CH
ECKSUM OF"CK"SHOULD BE"CH: STOP
5 NEXT AD: PRINT" READY FOR FLY?
 (Y/N/BREAK)"
6 IF INKEY$<>"Y" THEN 6
7 EXEC 8192

10 DATA 158,116,48,137,255,56,31
,20,189,169,40,48,141,2,128,206,
4,161,198,29,189,165,154,189,169
,162,189,169,118,204,48,48,3940
20 DATA 221,242,198,255,79,151,2
41,183,255,32,74,38,250,31,152,9
0,38,245,48,141,2,118,206,5,165,
198,18,189,165,154,223,136,4543
30 DATA 141,2,32,20,189,161,177,
129,3,39,181,129,48,35,245,129,5
7,34,241,128,48,151,195,57,79,15
1,194,151,240,134,9,151,3680
40 DATA 193,204,255,128,142,4,0,
167,128,140,4,33,38,249,231,128,
140,5,224,38,249,198,255,237,129
,140,5,255,35,249,48,31,4282
50 DATA 48,136,224,237,132,140,4
,32,34,246,198,175,142,4,240,231
,132,159,208,215,210,198,159,142
,4,80,231,132,159,211,215,213,48
91
60 DATA 142,4,22,204,51,58,237,1
29,204,48,48,237,132,142,4,5,237
,132,142,4,3,204,83,67,237,132,1
42,4,13,204,72,83,3426
70 DATA 237,132,220,242,253,4,15
,23,1,169,132,255,129,193,36,16,
129,129,36,16,129,65,36,4,134,10
,32,10,134,94,32,6,3053
80 DATA 134,8,32,2,134,9,151,192
,23,1,136,132,6,139,3,52,2,141,1
7,150,194,129,68,16,39,255,93,53
,2,74,129,0,2516
```

```
90 DATA 38,237,32,195,182,1,19,1
29,59,35,51,134,0,183,1,19,182,4
,25,74,183,4,25,129,47,38,35,134
,57,183,4,25,2464
100 DATA 182,4,24,74,183,4,24,12
9,47,38,19,134,53,183,4,24,182,4
,22,74,183,4,22,129,47,38,3,22,0
,242,222,195,2515
110 DATA 51,95,17,131,0,0,38,248
,214,210,158,208,150,192,141,76,
159,208,215,210,158,211,214,213,
189,161,193,129,84,16,39,0,4328
120 DATA 223,129,83,38,40,52,54,
141,2,32,18,206,5,230,48,141,1,5
8,198,18,189,165,154,223,136,23,
254,200,57,134,255,222,3729
130 DATA 136,167,196,51,95,17,13
1,5,229,38,246,53,54,129,0,39,2,
151,193,150,193,141,5,159,211,21
5,213,57,129,8,38,6,3457
140 DATA 31,18,48,31,32,30,129,9
,38,6,31,18,48,1,32,20,129,94,38
,7,31,18,48,136,224,32,9,129,10,
38,58,31,1554
150 DATA 18,48,136,32,141,1,57,1
66,132,129,128,39,5,231,164,141,
6,57,231,132,167,164,57,129,255,
38,3,31,33,57,129,159,3216
```

```
160 DATA 38,116,52,54,23,0,120,5
3,54,134,255,167,164,166,63,129,
128,38,7,49,63,167,164,31,33,57,
166,33,129,128,38,4,2823
170 DATA 49,33,32,241,166,168,22
4,129,128,38,5,49,168,224,32,229
,166,168,32,129,128,38,226,49,16
8,32,32,217,142,4,5,236,3687
180 DATA 132,92,193,58,39,4,237,
132,32,5,198,48,76,237,132,57,14
2,4,22,204,48,58,237,129,198,48,
237,132,134,68,151,194,3678
190 DATA 150,240,145,241,35,10,1
51,241,252,4,5,221,242,253,4,15,
189,33,107,22,254,2,31,33,129,17
5,16,39,255,132,57,189,3872
200 DATA 169,118,198,21,134,255,
183,255,32,74,38,250,90,38,245,1
2,240,32,169,134,0,71,142,1,22,1
98,3,166,133,90,169,133,3815
210 DATA 167,133,90,42,249,198,3
,108,133,38,3,90,42,249,57,82,79
,78,65,76,68,96,66,79,85,76,65,7
8,71,69,82,96,2913
220 DATA 80,82,69,83,69,78,84,83
,96,70,76,89,83,69,76,69,67,84,9
6,83,80,69,69,68,96,40,49,45,57,
41,255,53,2508
```

■ ■ ■

Program Listing. Fly!

```
           A1C1           00001 POLCAT  EQU   $A1C1          20B0 ED   84      00085         STD    ,X
           A1B1           00002 POLKEY  EQU   $A1B1          20B2 8E   0403    00086         LDX    #$403
           A976           00003 AUDON   EQU   $A976          20B5 CC   5343    00087         LDD    #$5343
           A9A2           00004 SELMUX  EQU   $A9A2          20B8 ED   00      00088         STD    0,X
           A59A           00005 TXTMOV  EQU   $A59A          20BA 8E   040D    00089         LDX    #$40D
           A928           00006 CLSCRN  EQU   $A928          20BD CC   4853    00090         LDD    #$4853
                          00007                              20C0 ED   00      00091         STD    0,X
  2000                    00008         ORG   $2000          20C2 DC   F2      00092         LDD    $F2
  2000 9E   74            00009 INIT    LDX   $74            20C4 FD   040F    00093         STD    $40F
  2002 30   89 FF38       00010         LEAX  $FF38,X        20C7 17   01A9    00094 CPUDIR  LBSR   RND
  2006 1F   14            00011         TFR   X,S            20CA 84   FF      00095         ANDA   #$FF
  2008 BD   A928          00012         JSR   CLSCRN         20CC 81   C1      00096         CMPA   #193
  200B 30   8D 0280       00013         LEAX  MESG,PCR       20CE 24   10      00097         BHS    CURLFT
  200F CE   04A1          00014         LDU   #$4A1          20D0 81   81      00098         CMPA   #129
  2012 C6   1D            00015         LDB   #$1D           20D2 24   1D      00099         BHS    CURRGT
  2014 BD   A59A          00016         JSR   TXTMOV         20D4 81   41      00100         CMPA   #65
  2017 BD   A9A2          00017         JSR   SELMUX         20D6 24   04      00101         BHS    CURUP
  201A BD   A976          00018         JSR   AUDON          20D8 86   0A      00102 CURDN   LDA    #$0A
  201D CC   3030          00019         LDD   #$3030         20DA 20   0A      00103         BRA    DIR
  2020 DD   F2            00020         STD   $F2            20DC 86   5E      00104 CURUP   LDA    #$5E
  2022 C6   FF            00021         LDB   #$FF           20DE 20   06      00105         BRA    DIR
  2024 4F                 00022         CLRA                 20E0 86   08      00106 CURLFT  LDA    #$08
  2025 97   F1            00023         STA   $F1            20E2 20   02      00107         BRA    DIR
  2027 B7   FF20          00024 OUV     STA   $FF20          20E4 86   09      00108 CURRGT  LDA    #$09
  202A 4A                 00025         DECA                 20E6 97   C0      00109 DIR     STA    $C0
  202B 26   FA            00026         BNE   OUV            20E8 17   0188    00110 DIRECT  LBSR   RND
  202D 1F   98            00027         TFR   B,A            20EB 84   06      00111         ANDA   #$06
  202F 5A                 00028         DECB                 20ED 8B   03      00112         ADDA   #$03
  2030 26   F5            00029         BNE   OUV            20EF 34   02      00113 NEXT    PSHS   A
  2032 30   8D 0276       00030         LEAX  SEL,PCR        20F1 8D   11      00114         BSR    MAIN
  2036 CE   05A5          00031         LDU   #$5A5          20F3 96   C2      00115         LDA    $C2
  2039 C6   12            00032         LDB   #$12           20F5 81   44      00116         CMPA   #$44
  203B BD   A59A          00033         JSR   TXTMOV         20F7 1027 FF5D    00117         LBEQ   BEGIN
  203E DF   88            00034         STU   $88            20FB 35   02      00118 DEC     PULS   A
  2040 8D   02            00035         BSR   POLL           20FD 4A           00119         DECA
  2042 20   14            00036         BRA   BEGIN          20FE 81   00      00120         CMPA   #$00
  2044 BD   A1B1          00037 POLL    JSR   POLKEY         2100 26   ED      00121         BNE    NEXT
  2047 81   03            00038         CMPA  #$03           2102 20   C3      00122         BRA    CPUDIR
  2049 27   B5            00039         BEQ   INIT           2104 B6   0113    00123 MAIN    LDA    $0113
  204B 81   30            00040         CMPA  #$30           2107 81   3B      00124         CMPA   #$3B
  204D 23   F5            00041         BLS   POLL           2109 23   33      00125         BLS    TIME
  204F 81   39            00042         CMPA  #$39           210B 86   00      00126         LDA    #$00
  2051 22   F1            00043         BHI   POLL           210D B7   0113    00127         STA    $0113
  2053 80   30            00044         SUBA  #$30           2110 B6   0419    00128         LDA    $0419
  2055 97   C3            00045         STA   $C3            2113 4A           00129         DECA
  2057 39                 00046         RTS                  2114 B7   0419    00130         STA    $0419
  2058 4F                 00047 BEGIN   CLRA                 2117 81   2F      00131         CMPA   #$2F
  2059 97   C2            00048         STA   $C2            2119 26   23      00132         BNE    TIME
  205B 97   F0            00049         STA   $F0            211B 86   39      00133         LDA    #$39
  205D 86   09            00050         LDA   #$09           211D B7   0419    00134         STA    $0419
  205F 97   C1            00051         STA   $C1            2120 B6   0418    00135         LDA    $0418
  2061 CC   FF80          00052         LDD   #$FF80         2123 4A           00136         DECA
  2064 8E   0400          00053         LDX   #$0400         2124 B7   0418    00137         STA    $0418
  2067 A7   80            00054 UP      STA   ,X+            2127 81   2F      00138         CMPA   #$2F
  2069 8C   0421          00055         CMPX  #$0421         2129 26   13      00139         BNE    TIME
  206C 26   F9            00056         BNE   UP             212B 86   35      00140         LDA    #$35
  206E E7   80            00057 CLS     STB   ,X+            212D B7   0418    00141         STA    $0418
  2070 8C   05E0          00058         CMPX  #$05E0         2130 B6   0416    00142         LDA    $0416
  2073 26   F9            00059         BNE   CLS            2133 4A           00143         DECA
  2075 C6   FF            00060         LDB   #$FF           2134 B7   0416    00144         STA    $0416
  2077 ED   81            00061 DOWN    STD   ,X++           2137 81   2F      00145         CMPA   #$2F
  2079 8C   05FF          00062         CMPX  #$5FF          2139 26   03      00146         BNE    TIME
  207C 23   F9            00063         BLS   DOWN           213B 16   00F2    00147         LBRA   TIME0
  207E 30   1F            00064         LEAX  -1,X           213E DE   C3      00148 TIME    LDU    $C3
  2080 30   88 E0         00065 SIDE    LEAX  -32,X          2140 33   5F      00149 DELAY   LEAU   -01,U
  2083 ED   84            00066         STD   ,X             2142 1183 0000    00150         CMPU   #$0000
  2085 8C   0420          00067         CMPX  #$420          2146 26   F8      00151         BNE    DELAY
  2088 22   F6            00068         BHI   SIDE           2148 D6   D2      00152         LDB    $D2
  208A C6   AF            00069         LDB   #$AF           214A 9E   D0      00153         LDX    $D0
  208C 8E   04F0          00070         LDX   #$4F0          214C 96   C0      00154         LDA    $C0
  208F E7   84            00071         STB   ,X             214E 8D   4C      00155         BSR    CPUMOV
  2091 9F   D0            00072         STX   $D0            2150 9F   D0      00156         STX    $D0
  2093 D7   D2            00073         STB   $D2            2152 D7   D2      00157         STB    $D2
  2095 C6   9F            00074         LDB   #$9F           2154 9E   D3      00158         LDX    $D3
  2097 8E   0450          00075         LDX   #$450          2156 D6   D5      00159         LDB    $D5
  209A E7   84            00076         STB   ,X             2158 BD   A1C1    00160         JSR    POLCAT
  209C 9F   D3            00077         STX   $D3            215B 81   54      00161         CMPA   #$54
  209E D7   D5            00078         STB   $D5            215D 1027 00DF    00162         LBEQ   HIGHS
  20A0 8E   0416          00079         LDX   #$416          2161 81   53      00163         CMPA   #$53
  20A3 CC   333A          00080         LDD   #$333A         2163 26   28      00164         BNE    COMPA
  20A6 ED   81            00081         STD   ,X++           2165 34   36      00165         PSHS   A,B,X,Y
  20A8 CC   3030          00082         LDD   #$3030         2167 8D   02      00166         BSR    PRTSEL
  20AB ED   84            00083         STD   ,X             2169 20   12      00167         BRA    ERASE
  20AD 8E   0405          00084         LDX   #$405          216B CE   05E6    00168 PRTSEL  LDU    #$5E6
```

```
216E 30  8D 013A  00169       LEAX   SEL,PCR
2172 C6  12        00170       LDB    #$12
2174 BD  A928      00171       JSR    TXTMOV
2177 DF  88        00172       STU    $88
2179 17  FEC8      00173       LBSR   POLL
217C 39            00174       RTS
217D 86  FF        00175 ERASE LDA    #$FF
217F DE  88        00176       LDU    $88
2181 A7  40        00177 BACKSP STA   0,U
2183 33  5F        00178       LEAU   -1,U
2185 1183 05E5     00179       CMPU   #$5E5
2189 26  F6        00180       BNE    BACKSP
218B 35  36        00181       PULS   A,B,X,Y
218D 81  00        00182 COMPA CMPA   #$00
218F 27  02        00183       BEQ    PLYDIR
2191 97  C1        00184       STA    $C1
2193 96  C1        00185 PLYDIR LDA   $C1
2195 8D  05        00186       BSR    CPUMOV
2197 9F  D3        00187       STX    $D3
2199 D7  D5        00188       STB    $D5
219B 39            00189       RTS
219C 81  08        00190 CPUMOV CMPA  #$08
219E 26  06        00191       BNE    CPURGT
21A0 1F  12        00192       TFR    X,Y
21A2 30  1F        00193       LEAX   -01,Y
21A4 20  1E        00194       BRA    CHK1
21A6 81  09        00195 CPURGT CMPA  #$09
21A8 26  06        00196       BNE    CPUUP
21AA 1F  12        00197       TFR    X,Y
21AC 30  01        00198       LEAX   01,X
21AE 20  14        00199       BRA    CHK1
21B0 81  5E        00200 CPUUP CMPA   #$5E
21B2 26  07        00201       BNE    CPUDN
21B4 1F  12        00202       TFR    X,Y
21B6 30  88 E0     00203       LEAX   -$20,X
21B9 20  09        00204       BRA    CHK1
21BB 81  0A        00205 CPUDN CMPA   #$0A
21BD 26  3A        00206       BNE    RTN
21BF 1F  12        00207       TFR    X,Y
21C1 30  88 20     00208       LEAX   $20,X
21C4 8D  01        00209 CHK1  BSR    CHCK
21C6 39            00210       RTS
21C7 A6  00        00211 CHCK  LDA    0,X
21C9 81  80        00212       CMPA   #$80
21CB 27  05        00213       BEQ    EXCH
21CD E7  20        00214       STB    0,Y
21CF 8D  06        00215       BSR    CKSIDE
21D1 39            00216       RTS
21D2 E7  00        00217 EXCH  STB    0,X
21D4 A7  20        00218       STA    0,Y
21D6 39            00219       RTS
21D7 81  FF        00220 CKSIDE CMPA  #$FF
21D9 26  03        00221       BNE    OPPONT
21DB 1F  21        00222       TFR    Y,X
21DD 39            00223       RTS
21DE 81  9F        00224 OPPONT CMPA  #$9F
21E0 26  74        00225       BNE    HITCHK
21E2 34  36        00226 HIT   PSHS   Y,X,B,A
21E4 17  0078      00227       LBSR   SOUND
21E7 35  36        00228       PULS   Y,X,B,A
21E9 86  FF        00229       LDA    #$FF
21EB A7  20        00230       STA    0,Y
21ED A6  3F        00231       LDA    -01,Y
21EF 81  80        00232       CMPA   #$80
21F1 26  07        00233       BNE    SPACE
21F3 31  3F        00234       LEAY   -01,Y
21F5 A7  20        00235 SWAP  STA    0,Y
21F7 1F  21        00236       TFR    Y,X
21F9 39            00237 RTN   RTS
21FA A6  21        00238 SPACE LDA    01,Y
21FC 81  80        00239       CMPA   #$80
21FE 26  04        00240       BNE    SPACE1
2200 31  21        00241       LEAY   01,Y
2202 20  F1        00242       BRA    SWAP
2204 A6  A8 E0     00243 SPACE1 LDA   -$20,Y
2207 81  80        00244       CMPA   #$80
2209 26  05        00245       BNE    SPACE2
220B 31  A8 E0     00246       LEAY   -$20,Y
220E 20  E5        00247       BRA    SWAP
2210 A6  A8 20     00248 SPACE2 LDA   $20,Y
2213 81  80        00249       CMPA   #$80
2215 26  E2        00250       BNE    RTN
2217 31  A8 20     00251       LEAY   $20,Y
221A 20  D9        00252       BRA    SWAP
221C 8E  0405      00253 SCORE LDX    #$0405
221F EC  00        00254       LDD    0,X
2221 5C            00255       INCB
2222 C1  3A        00256       CMPB   #$3A
2224 27  04        00257       BEQ    RESET0
2226 ED  00        00258       STD    0,X
2228 20  05        00259       BRA    COUNT1
222A C6  30        00260 RESET0 LDB   #$30
222C 4C            00261       INCA
222D ED  00        00262       STD    0,X
222F 39            00263 COUNT1 RTS
2230 8E  0416      00264 TIME0 LDX    #$0416
2233 CC  303A      00265       LDD    #$303A
2236 ED  81        00266       STD    ,X++
2238 C6  30        00267       LDB    #$30
223A ED  00        00268       STD    0,X
223C 86  44        00269       LDA    #$44
223E 97  C2        00270       STA    $C2
2240 96  F0        00271 HIGHS LDA    $F0
2242 91  F1        00272       CMPA   $F1
2244 23  0A        00273       BLS    RETRY
2246 97  F1        00274       STA    $F1
2248 FC  0405      00275       LDD    0405
224B DD  F2        00276       STD    $F2
224D FD  040F      00277       STD    $40F
2250 BD  216B      00278 RETRY JSR    PRTSEL
2253 16  FE02      00279       LBRA   BEGIN
2256 1F  21        00280 HITCHK TFR   Y,X
2258 81  AF        00281       CMPA   #$AF
225A 1027 FF84     00282       LBEQ   HIT
225E 39            00283       RTS
225F BD  A976      00284 SOUND JSR    AUDON
2262 C6  15        00285       LDB    #$15
2264 86  FF        00286 LENTH LDA    #$FF
2266 B7  FF20      00287 TONE  STA    $FF20
2269 4A            00288       DECA
226A 26  FA        00289       BNE    TONE
226C 5A            00290       DECB
226D 26  F5        00291       BNE    LENTH
226F 0C  F0        00292       INC    $F0
2271 20  A9        00293       BRA    SCORE
2273 86  00        00294 RND   LDA    #0
2275 47            00295       ASRA
2276 8E  0116      00296       LDX    #$116
2279 C6  03        00297       LDB    #$03
227B A6  85        00298       LDA    B,X
227D 5A            00299       DECB
227E A9  85        00300 LOOP1 ADCA   B,X
2280 A7  85        00301       STA    B,X
2282 5A            00302       DECB
2283 2A  F9        00303       BPL    LOOP1
2285 C6  03        00304       LDB    #$03
2287 6C  85        00305 LOOP2 INC    B,X
2289 26  03        00306       BNE    RTS1
228B 5A            00307       DECB
228C 2A  F9        00308       BPL    LOOP2
228E 39            00309 RTS1  RTS
228F 524F4E414C44  00310 MESG  FCC    /RONALD/
2295 60            00311       FCB    $60
2296 424F554C414E  00312       FCC    /BOULANGER/
     474552
         (Note- FCC output artificially compressed -Ed.)
229F 60            00313       FCB    $60
22A0 50524553454E  00314       FCC    /PRESENTS/
     5453
22A8 60            00313       FCB    $60
22A9 464C59        00316       FCC    /FLY/
22AC 53454C454354  00317 SEL   FCC    /SELECT/
22B2 60            00318       FCB    $60
22B3 5350454544    00319       FCC    /SPEED/
22B8 60            00320       FCB    $60
22B9 28312D3939    00321       FCC    /(1-9)/
     2000          00322       END    INIT
00000 TOTAL ERRORS
```

AUDON	A976	COMPA	218D	CURLFT	20E0	ERASE	217D	LOOP2	2287	POLKEY	A1B1	SCORE	221C	SWAP	21F5
BACKSP	2181	COUNT1	222F	CURRGT	20E4	EXCH	21D2	MAIN	2104	POLL	2044	SEL	22AC	TFR	21DB
BEGIN	2058	CPUDIR	20C7	CURUP	20DC	HIGHS	2240	MESG	228F	PRTSEL	216B	SELMUX	A9A2	TIME	213E
CHCK	21C7	CPUDN	21BB	DEC	20FB	HIT	21E2	NEXT	20EF	RESET0	222A	SIDE	2080	TIME0	2230
CHK1	21C4	CPUMOV	219C	DELAY	2140	HITCHK	2256	OPPONT	21DE	RETRY	2250	SOUND	225F	TONE	2266
CKSIDE	21D7	CPURGT	21A6	DIR	20E6	INIT	2000	OUV	2027	RND	2273	SPACE	21FA	TXTMOV	A928
CLS	206E	CPUUP	21B0	DIRECT	20E8	LENTH	2264	PLYDIR	2193	RTN	21F9	SPACE1	2204	UP	2067
CLSCRN	A928	CURDN	20D8	DOWN	2077	LOOP1	227E	POLCAT	A1C1	RTS1	228E	SPACE2	2210		

AND, OR, NOT

*Boolean functions, and
what they <u>really</u> mean!*

4K Color
Basic

Tom Fish

by Jon J. Schild

BESIDES THE NORMAL operators for addition, subtraction, multiplication, division, and exponentiation, we have three logical, or Boolean, operators in Basic: AND, OR, and NOT. In most versions of Basic (including Color and Extended Color) these may be used to combine conditions in an If statement, as in IF J>0 AND K=1 THEN PRINT "YES". Anyone who sees this realizes that both conditions must be true before the word YES will be printed. Likewise, IF J>0 OR K=1 THEN PRINT "YES" means that if either condition is true (or if both are true) the Print statement will be executed.

Boolean operators combine conditions as stated in Basic rather than as stated in English. In English it is quite permissible to say "If A is 2 or 3, add 1 to C." This is not the same as IF A=2 OR 3 THEN C=C+1 as many novice programmers have discovered. In some versions of Basic this statement would be an error. In

Color Computer Basic it is not an error, but it does not do what its author intended.

A similar problem occurs with NOT. It negates a condition stated in Basic, not as stated in English. The English statement "If A is not less than 10, add 1 to C" is not translated as IF A NOT < 10 THEN C=C+1 but as IF NOT A<10 THEN C=C+1. The condition (A<10) must be negated as a whole.

Boolean operators were originally defined on a bit-by-bit basis and can still be used that way in Color Computer Basic. Since they are operators, it is possible to set up a table to show their possible argument values and their results (Table 1).

Each bit of the two operands takes part in the operation as shown. The following program illustrates the operation of AND and OR on whole operands.

```
10 FOR X = 0 TO 15 : FOR Y = 0 TO 15
20 PRINT X; Y, X and Y; X OR Y
30 NEXT Y
40 PRINT
50 NEXT X
```

Notice that 3 OR 5 is 7 (in bits: 011 OR 101 = 111) because each bit in one operand is ORed with the corresponding bit in the other. Also, note 3 AND 5 is 1 (011 AND 101 = 001) because the bits are ANDed together in corresponding pairs.

Nice. But how does that relate to the AND and OR in If statements? They are exactly the same thing. All we need do is define the terms *true* and *false* as they relate to conditions and If statements. The required definitions are:

- True relates to *conditions* the same as −1 (all bits set to 1). Really! Try it. Just type PRINT 3=3. You will get −1.
- False relates to *conditions* the same as zero (all bits set to 0). You can demonstrate that with PRINT 2+2=5.
- True relates to *If statements* as anything but zero. The Then part of the statement is executed if the condition equals any value except zero.
- False relates to *If statements* as zero, the same as for conditions.

In the normal case we would have something like: IF J>0 OR K=1 THEN PRINT "YES" ELSE PRINT "NO". The condition J>0 is evaluated and yields either a −1 or a zero. The K=1 condition likewise yields either −1 or zero. These two values are then ORed together, and if the result is other than zero, YES is printed. If the result is zero, NO is printed.

Let's use these definitions to see why IF A=2 OR 3 THEN C=C+1 doesn't do what was expected. According to Tandy's documentation, relational operators (equals, less than, greater than, less than or equal to, greater than or equal to, and not equal to are executed before Boolean operators (AND, OR, NOT). A = 2 is evaluated first, giving as its result either −1 or zero depending on the current value of A. The result is then ORed with three. This can never result in a zero value, so C = C + 1 is always executed no matter what value A has.

The Boolean operators can be used in normal Let statements also, which means they can be used in functions defined by the DEF FN statements in Extended Color Basic. That is useful, if you know how to use them. For instance, I wrote a program that involved PEEKing characters from the Color Computer's screen memory. It didn't work. I found that because of the internal character generator in the Video Display Generator (VDG) chip, the

X	Y	X AND Y	X OR Y	NOT X
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Table 1. Argument Values and Results

Character Series	PEEK/POKE Code	ASCII Code
@ thru underscore (normal)	&H40–&H5F	&H40–&H5F
@ thru underscore (reversed)	&H00–&H1F	&H60–&H7F
space through ? (normal)	&H60–&H7F	&H20–&H3F
space through ? (reversed)	&H20–&H3F	N/A

Table 2. Character Series and Codes

code stored in screen memory doesn't always match the normal ASCII code sent via Print statements. CLS : PRINT CHR$(48) : PRINT @ 64 prints a zero at the upper left corner of the screen, but CLS : POKE 1024, 48 : PRINT @ 64 prints a reversed zero. Look at Table 2.

It is necessary to convert back and forth in several places. The obvious way to handle this is by using either a function or a GOSUB command to do the conversion. Since Extended Basic lets us define functions, that seems the easiest way to go. Without Extended Basic it would be necessary to use GOSUBs. (It is also possible to do the conversions in assembly language.) I chose to define one function for each conversion. I wrote the two functions below, FNA to convert to ASCII from PEEK/POKE code, and FNP to convert to PEEK/POKE code from ASCII. I will state the functions first, then show how they relate to the discussion in the early part of the article.
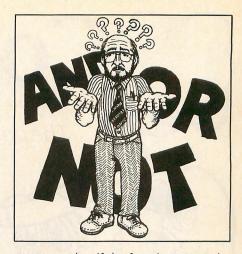
```
5 DEF FNA(X) = ((((X<&H20) AND &H60) OR
((X<&H40) AND (X>&H1F) AND &HE0) OR
((X<&H80) AND (X>&H5F) AND &HC0)) + X) AND
&HFF
10 DEF FNP(X) = ((((X<&H20) AND &H20) OR
((X<&H40) AND (X>&H1F) AND &H40) OR
((X<&H80) AND (X>&H5F) AND &HA0)) + X) AND
&HFF
```

These functions look quite forbidding, so let's examine them piece by piece, beginning with FNA.

(X<&H20) AND &H60) If the PEEK/POKE code (PPc) is less than &H20 (that is, in the range &H00 – &H1F) it needs to be converted to the range &H60 – &H7F. This requires adding &H60 to whatever the code may be. If X is any value in this range, the condition (X<&H20) will be true. That is, it will evaluate to −1 (all bits 1). So ((X<&H20) AND &H60) will evaluate to &H60. If X is a value outside this range, (X<&H20) will be false, or zero, and ((X<&H20) AND &H60) will evaluate to zero.

((X<&H40) AND (X>&H1F) AND &HE0) If the PPc is in the range &H20 through &H3F we need to convert it to the range &H00 through &H1F. This range cannot actually be put on the screen with a Print

statement, but if the functions are to be inverses of each other all codes must convert to something. *This requires* subtracting &H20, which is the same (for an eight-bit byte) as adding &HE0 and then throwing away the high-order bit. We'll worry about that high-order bit later. For a value in the range &H20 through &H3F, both (X<&H40) and (X>&H1F) will evaluate to true (−1) while for any X value outside this range one condition or the other will evaluate to false (0). When the two are ANDed together, a true (−1) will be produced only if both conditions are met. −1 AND &HE0 will give &HE0. If the X value is outside this range, we have 0 AND &HE0, which gives a zero.

((X<&H80) AND (X>&H5F) AND &HC0) In a similar fashion to the preceding part, this will produce &HC0 for an X value in the range &H60 through &H7F or zero for any value outside this range.

These three values (two of which will be zero) are ORed together (they could have been added with the same result) and the result is added to the original X value. Two of the ranges will produce an extra high-order bit which needs to be discarded. The AND &HFF at the end of the function will keep only the eight low-order bits, getting rid of the troublesome high-order bit we didn't want. Voila! Whatever PPc was in X to start with, the function now returns the proper adjusted value. Note that two of the PPc ranges, &H40 through &H5F and &H80 through &HFF, will fail all tests and cause a zero to be added to X at the end. That is because those two ranges are the same in PPc and ASCII and thus do not need to be converted.

FNP will work the same way as FNA to perform the reverse conversion.

To use these two in non-Extended Basic, replace DEF FNA(X) or DEF FNP(X) on the left of the equal sign by Y or some other variable *name*, add :RETURN to the end, and use them as GOSUBs. ■ ■ ■

Color Burner
Part III

The <u>conclusive</u> conclusion to Kitsz's EPROM Burner ... finally!

by Dennis Kitsz

EDTASM +

16K Extended Color Basic

Partial code in speech bubble:
```
      00120 *
      00130 *
0600  00140 VIDEX1   F
E00   00150 VIDEX2   F
7F40  00160 ENDBN1   F
7F80  00170 ENDBN2   F
7F40  00180 ENDBAN
0000  00190 NODOTS
02    00200 DUBLER
10    00210 NOCHRS
0000  00220 STOREB
0000  00230 STOREY
      00240 *
7F00  00250 BANTBL
      00260 ASCII
```

Editor's note: This month we present the assembly language program for Dennis's Color Burner (Part I — March 1984, Part II — May 1984). This is the last bit of information you'll need to use the Color Burner. We apologize that space considerations made this delay necessary. The package is finally complete!

TAPE LOADER MAY ISSUE

Program Listing 2. Assembly Language Burner

```
                00100 ************************
                00110 * COLOR BURNER DRIVER 1.0
                00120 * BY DENNIS BATHORY KITSZ
                00130 * COPYRIGHT (C) 1984 BY
                00140 * DENNIS BATHORY KITSZ
                00150 ************************
                00160 *
                00170 *
3D00            00180        ORG       $3D00
                00190 *
                00200 * OFFSET (0 4 8 12 16 20
                00210 * 24 28) PLACED BY BASIC
                00220 * TO INDICATE EPROM TYPE
                00230 *
3D00  00        00240 OFFSET FCB       $00
3D01  0000      00250 STTADR FDB       $0000
3D03  4000      00260 STTDAT FDB       $4000
3D05  0800      00270 ENDADR FDB       $0800
                00280 *
                00290 * THE TABLES BELOW HOLD
                00300 * THE PROGRAMMING DETAILS
                00310 * FOR EIGHT EPROMS. FIXED
                00320 * TABLES ARE PROVIDED FOR
                00330 * 2716, 2732, 2764 AND
                00340 * 27128 EPROMS.  OTHER
                00350 * TYPES ARE FILLED IN BY
                00360 * THE BASIC DRIVER.
                00370 *
      3D07      00380 TAB16  EQU       *
3D07  03        00390 O2716  FCB       $03
3D08  C0        00400 R2716  FCB       $C0
3D09  91        00410 S2716  FCB       $91
3D0A  93        00420 P2716  FCB       $93
                00430 *
      3D0B      00440 TAB32  EQU       *
3D0B  02        00450 O2732  FCB       $02
3D0C  C0        00460 R2732  FCB       $C0
3D0D  90        00470 S2732  FCB       $90
3D0E  80        00480 P2732  FCB       $80
                00490 *
      3D0F      00500 TAB64  EQU       *
3D0F  03        00510 O2764  FCB       $03
3D10  C4        00520 R2764  FCB       $C4
3D11  A4        00530 S2764  FCB       $A4
3D12  A0        00540 P2764  FCB       $A0
                00550 *
      3D13      00560 TAB128 EQU       *
3D13  03        00570 O27128 FCB       $03
3D14  C4        00580 R27128 FCB       $C4
3D15  A4        00590 S27128 FCB       $A4
```

```
3D16  A0        00600 P27128 FCB       $A0
                00610 *
      3D17      00620 TABSP1 EQU       *
3D17  00        00630 OSP1   FCB       $00
3D18  00        00640 RSP1   FCB       $00
3D19  00        00650 SSP1   FCB       $00
3D1A  00        00660 PSP1   FCB       $00
                00670 *
      3D1B      00680 TABSP2 EQU       *
3D1B  00        00690 OSP2   FCB       $00
3D1C  00        00700 RSP2   FCB       $00
3D1D  00        00710 SSP2   FCB       $00
3D1E  00        00720 PSP2   FCB       $00
                00730 *
      3D1F      00740 TABSP3 EQU       *
3D1F  00        00750 OSP3   FCB       $00
3D20  00        00760 RSP3   FCB       $00
3D21  00        00770 SSP3   FCB       $00
3D22  00        00780 PSP3   FCB       $00
                00790 *
      3D23      00800 TABSP4 EQU       *
3D23  00        00810 OSP4   FCB       $00
3D24  00        00820 RSP4   FCB       $00
3D25  00        00830 SSP4   FCB       $00
3D26  00        00840 PSP4   FCB       $00
                00850 *
                00860 * THIS ROUTINE CONFIGURES
                00870 * PORT A AS AN INPUT PORT
                00880 *
3D27  8E FF40   00890 CONAIN LDX       #$FF40
3D2A  C6 04     00900        LDB       #$04
3D2C  4F        00910        CLRA
3D2D  A7 01     00920        STA       1,X
3D2F  A7 84     00930        STA       ,X
3D31  E7 01     00940        STB       1,X
3D33  39        00950        RTS
                00960 *
                00970 * THIS ROUTINE CONFIGURES
                00980 * THE A PORT AS OUTPUT
                00990 *
3D34  8E FF40   01000 CONAOT LDX       #$FF40
3D37  C6 04     01010        LDB       #$04
3D39  4F        01020        CLRA
3D3A  A7 01     01030        STA       1,X
3D3C  4A        01040        DECA
3D3D  A7 84     01050        STA       ,X
3D3F  E7 01     01060        STB       1,X
3D41  39        01070        RTS
                01080 *
```

```
                      01090 * THIS ROUTINE CONFIGURES
                      01100 * PORTS B, C AND D AS
                      01110 * OUTPUTS.  B, C AND D
                      01120 * ARE USED TO PROVIDE A
                      01130 * BYTE OF DATA AND AN
                      01140 * ADDRESS TO THE EPROM
                      01150 *
3D42 C6   04          01160 CONBCD LDB     #$04
3D44 8E   FF40        01170        LDX     #$FF40
3D47 4F               01180        CLRA
3D48 A7   03          01190        STA     3,X
3D4A 4A               01200        DECA
3D4B A7   02          01210        STA     2,X
3D4D E7   03          01220        STB     3,X
3D4F 4F               01230        CLRA
3D50 A7   05          01240        STA     5,X
3D52 4A               01250        DECA
3D53 A7   04          01260        STA     4,X
3D55 E7   05          01270        STB     5,X
3D57 4F               01280        CLRA
3D58 A7   07          01290        STA     7,X
3D5A 4A               01300        DECA
3D5B A7   06          01310        STA     6,X
3D5D E7   07          01320        STB     7,X
3D5F 39               01330        RTS
                      01340 *
                      01350 * THIS ROUTINE USES THE
                      01360 * OFFSET POKED BY BASIC
                      01370 * TO DETERMINE THE EPROM
                      01380 * TYPE AND SELECT THE
                      01390 * CORRECT TABLE FOR USE
                      01400 * BY PROGRAMMING SOFTWARE
                      01410 *
3D60 34   10          01420 ETYPE  PSHS    X
3D62 8E   3D07        01430        LDX     #TAB16
3D65 B6   3D00        01440        LDA     OFFSET
3D68 31   86          01450        LEAY    A,X
3D6A 35   10          01460        PULS    X
3D6C 39               01470        RTS
                      01480 *
                      01490 * THIS ROUTINE DISPLAYS
                      01500 * MESSAGES DURING READ,
                      01510 * PROGRAM, VERIFY AND
                      01520 * ERASE CHECK.
                      01530 *
3D6D 108E 0460        01540 MESSER LDY     #$0460
3D71 C6   10          01550        LDB     #$10
3D73 A6   80          01560 MLOOP1 LDA     ,X+
3D75 A7   A0          01570        STA     ,Y+
3D77 5A               01580        DECB
3D78 26   F9          01590        BNE     MLOOP1
3D7A 39               01600        RTS
                      01610 *
                      01620 * THIS ROUTINE PROVIDES
                      01630 * A SHORT DELAY TO ASSURE
                      01640 * THAT CONTROL SIGNALS
                      01650 * ARE PROPERLY RECEIVED
                      01660 * BY BOTH PIA AND EPROM
                      01670 *
3D7B 34   04          01680 DELAY1 PSHS    B
3D7D C6   04          01690        LDB     #$04
3D7F 5A               01700 LOOP1  DECB
3D80 26   FD          01710        BNE     LOOP1
3D82 35   04          01720        PULS    B
3D84 39               01730        RTS
                      01740 *
                      01750 * THIS ROUTINE CREATES A
                      01760 * 50 MS DELAY REQUIRED
                      01770 * FOR THE PROGRAMMING
                      01780 * PROCESS.  THE ACTUAL
                      01790 * DELAY CREATED HERE IS
                      01800 * APPROXIMATELY 50.1 MS
                      01810 *
3D85 34   06          01820 D50MS  PSHS    D
3D87 CC   18F8        01830        LDD     #$18F8
3D8A 83   0001        01840 LOOP50 SUBD    #$0001
3D8D 26   FB          01850        BNE     LOOP50
3D8F 35   06          01860        PULS    D
3D91 39               01870        RTS
                      01880 *
                      01890 * THIS ROUTINE DISPLAYS
                      01900 * A 16-BIT ADDRESS IN HEX
                      01910 * BY CALLING THE BYTE
                      01920 * DISPLAY ROUTINE TWICE
                      01930 *
3D92 34   30          01940 DISADR PSHS    X,Y
3D94 108E 0470        01950        LDY     #$0470
3D98 8D   07          01960        BSR     SHOWAD
3D9A 1F   98          01970        TFR     B,A
3D9C 8D   03          01980        BSR     SHOWAD
3D9E 35   30          01990        PULS    X,Y
3DA0 39               02000        RTS
                      02010 *
                      02020 * THIS ROUTINE CONVERTS
                      02030 * ONE BYTE FROM BINARY
                      02040 * TO TWO HEXADECIMAL
                      02050 * DIGITS AND DISPLAYS
                      02060 * THEM ON THE SCREEN.
                      02070 *
3DA1 34   02          02080 SHOWAD PSHS    A
3DA3 44               02090        LSRA
3DA4 44               02100        LSRA
3DA5 44               02110        LSRA
3DA6 44               02120        LSRA
3DA7 8D   0B          02130        BSR     CONVRT
3DA9 A7   A0          02140        STA     ,Y+
3DAB 35   02          02150        PULS    A
3DAD 84   0F          02160        ANDA    #$0F
3DAF 8D   03          02170        BSR     CONVRT
3DB1 A7   A0          02180        STA     ,Y+
3DB3 39               02190        RTS
3DB4 81   0A          02200 CONVRT CMPA    #$0A
3DB6 24   03          02210        BCC     LETTER
3DB8 8B   70          02220        ADDA    #$70
3DBA 39               02230        RTS
3DBB 8B   37          02240 LETTER ADDA    #$37
3DBD 39               02250        RTS
                      02260 *
                      02270 * THE EPROM PROGRAMMING
                      02280 * ROUTINE BEGINS HERE.
                      02290 * IT CONSISTS OF PORT
                      02300 * CONFIGURATION, SETTING
                      02310 * UP ADDRESS AND DATA TO
                      02320 * THE EPROM, AND ISSUING
                      02330 * PROGRAMMING PULSES.
                      02340 *
3DBE 8E   3F28        02350 PROGMX LDX     #MESSG1
3DC1 17   FFA9        02360        LBSR    MESSER
                      02370 *
                      02380 * THIS ROUTINE CONFIGURES
                      02390 * PORT A FOR OUTPUT TO
                      02400 * EPROM, PORTS B C D FOR
                      02410 * OUTPUT TO EPROM.
                      02420 *
3DC4 17   FF6D        02430 CONFIG LBSR    CONAOT
3DC7 17   FF78        02440        LBSR    CONBCD
                      02450 *
                      02460 * D AND Y REGISTERS ARE
                      02470 * POINTED TO STARTING
                      02480 * ADDRESS OF THE EPROM
                      02490 * AND STARTING DATA IN
                      02500 * MEMORY TO PROGRAM.
                      02510 *
3DCA FC   3D01        02520        LDD     STTADR
3DCD 10BE 3D03        02530        LDY     STTDAT
                      02540 *
                      02550 * THIS ROUTINE LATCHES
                      02560 * DATA INTO THE LOWER
                      02570 * AND UPPER HALVES OF THE
                      02580 * EPROM'S ADDRESS, GETS
                      02590 * THE DATA FROM MEMORY,
                      02600 * AND LATCHES THAT TO THE
                      02610 * EPROM THROUGH THE PIA.
                      02620 *
3DD1 E7   02          02630 AGAIN  STB     2,X
3DD3 A7   04          02640        STA     4,X
3DD5 34   06          02650        PSHS    D
3DD7 A6   A0          02660        LDA     ,Y+
3DD9 A7   84          02670        STA     ,X
                      02680 *
                      02690 * THE ADDRESS AND DATA
                      02700 * ARE DISPLAYED FOR THE
                      02710 * USER AS THE PROGRAMMING
                      02720 * GOES ON.
                      02730 *
3DDB 34   20          02740        PSHS    Y         *
3DDD 108E 0478        02750        LDY     #$0478
3DE1 17   FFBD        02760        LBSR    SHOWAD
3DE4 35   20          02770        PULS    Y
3DE6 35   06          02780        PULS    D
3DE8 34   06          02790        PSHS    D
3DEA 17   FFA5        02800        LBSR    DISADR
3DED 35   06          02810        PULS    D
3DEF 34   06          02820        PSHS    D
                      02830 *
                      02840 * THE ACTUAL PROGRAMMING
                      02850 * TAKES PLACE IN THIS
                      02860 * ROUTINE.  INTERRUPTS
                      02870 * ARE DISABLED TO KEEP
                      02880 * THE PROGRAMMING PULSE
                      02890 * OF THE CORRECT LENGTH,
                      02900 * THE EPROM TYPE IS
                      02910 * ESTABLISHED, AND A
                      02920 * SEQUENCE OF OFF-READ-
                      02930 * GET SET-PULSE-GET SET-
```

```
                     02940 * READ-OFF IS FOLLOWED.                              03380 *
                     02950 *                               3E33 8E   3F88        03390          LDX    #MESSG7
3DF1 1A   50         02960 PROGRM ORCC    #$50            3E36 C6   0D          03400          LDB    #13
3DF3 34   20         02970        PSHS    Y                3E38 17   FF32        03410          LBSR   MESSER
3DF5 17   FF68       02980        LBSR    ETYPE           3E3B 39             03420          RTS
3DF8 A6   A4         02990        LDA     ,Y                                    03430 *
3DFA A7   06         03000        STA     6,X                                   03440 * THE CHECK FOR AN ERASED
3DFC 17   FF7C       03010        LBSR    DELAY1                                03450 * EPROM BEGINS HERE.  AN
3DFF A6   21         03020        LDA     1,Y                                   03460 * ERASED EPROM CONTAINS
3E01 A7   06         03030        STA     6,X                                   03470 * ALL FF (255) AS SENT
3E03 17   FF75       03040        LBSR    DELAY1                                03480 * FROM THE FACTORY, OR
3E06 A6   22         03050        LDA     2,Y                                   03490 * AFTER ERASURE UNDER
3E08 A7   06         03060        STA     6,X                                   03500 * ULTRAVIOLET LIGHT. THIS
3E0A 17   FF6E       03070        LBSR    DELAY1                                03510 * ROUTINE READS THE VALUE
3E0D A6   23         03080        LDA     3,Y                                   03520 * IN THE EPROM.
3E0F A7   06         03090        STA     6,X                                   03530 *
3E11 17   FF71       03100        LBSR    D50MS           3E3C 8E   3F48        03540 ERASEX LDX    #MESSG3
3E14 A6   22         03110        LDA     2,Y              3E3F 17   FF2B        03550          LBSR   MESSER
3E16 A7   06         03120        STA     6,X                                   03560 *
3E18 17   FF60       03130        LBSR    DELAY1                                03570 * SETUP WITH PORT A AS
3E1B A6   21         03140        LDA     1,Y                                   03580 * INPUT AND PORTS B, C
3E1D A7   06         03150        STA     6,X                                   03590 * AND D AS OUTPUT FOLLOWS
3E1F 17   FF59       03160        LBSR    DELAY1                                03600 *
3E22 A6   A4         03170        LDA     ,Y              3E42 17   FEE2        03610          LBSR   CONAIN
3E24 A7   06         03180        STA     6,X              3E45 17   FEFA        03620          LBSR   CONBCD
                     03190 *                               3E48 17   FF15        03630          LBSR   ETYPE
                     03200 * ONE BYTE HAS BEEN            3E4B A6   21          03640          LDA    1,Y
                     03210 * BURNED; A CHECK IS MADE      3E4D A7   06          03650          STA    6,X
                     03220 * TO SEE IF THE PROCESS                               03660 *
                     03230 * IS COMPLETE.  IF NOT,                               03670 * D IS POINTED TO THE
                     03240 * THE NEXT BYTE IS TAKEN                              03680 * STARTING ADDRESS IN THE
                     03250 * FROM MEMORY FOR BURNING                             03690 * EPROM
                     03260 * INTO THE EPROM.                                     03700 *
                     03270 *                               3E4F FC   3D01        03710          LDD    STTADR
3E26 35   20         03280        PULS    Y                                      03720 *
3E28 35   06         03290        PULS    D                                      03730 * THE LSB AND MSB OF THE
3E2A C3   0001       03300        ADDD    #$0001                                 03740 * ADDRESS ARE LATCHED
3E2D 10B3 3D05       03310        CMPD    ENDADR                                 03750 * INTO THE EPROM, AND
3E31 26   9E         03320        BNE     AGAIN                                  03760 * THE ADDRESS AND DATA
                     03330 *                                                     03770 * ARE DISPLAYED FOR THE
                     03340 * IF THE BURNING IS DONE,                             03780 * USER.
                     03350 * A MESSAGE IS DISPLAYED                              03790 *
                     03360 * AND CONTROL IS RETURNED      3E52 E7   02          03800 AGIN1  STB    2,X
                     03370 * TO BASIC.                    3E54 A7   04          03810          STA    4,X
```

```
3E56 34  06       03820        PSHS    D
3E58 17  FF37      03830        LBSR    DISADR
3E5B A6  84        03840        LDA     ,X
3E5D 34  22        03850        PSHS    Y,A
3E5F 108E 0478     03860        LDY     #$0478
3E63 17  FF3B      03870        LBSR    SHOWAD
3E66 35  22        03880        PULS    A,Y
3E68 81  FF        03890        CMPA    #$FF
3E6A 35  06        03900        PULS    D
3E6C 26  0E        03910        BNE     EFAULT
3E6E C3  0001      03920        ADDD    #$0001
3E71 10B3 3D05     03930        CMPD    ENDADR
3E75 26  DB        03940        BNE     AGIN1
                   03950 *
                   03960 * A CORRECT ERASURE OR
                   03970 * A FAILED ERASURE IS
                   03980 * DISPLAYED FOR THE USER.
                   03990 *
3E77 8E  3F98      04000        LDX     #MESSG8
3E7A 20  5A        04010        BRA     OUT1
                   04020 *
3E7C 8E  3F68      04030 EFAULT LDX     #MESSG5
3E7F 20  55        04040        BRA     OUT1
                   04050 *
                   04060 * THE VERIFY ROUTINE
                   04070 * (COMPARISON OF EPROM
                   04080 * WITH MEMORY CONTENTS)
                   04090 * BEGINS HERE.  THE SETUP
                   04100 * OF PORT A AS INPUT AND
                   04110 * PORTS B, C AND D AS
                   04120 * OUTPUT, PLUS EPROM
                   04130 * TYPING, IS DONE HERE.
                   04140 *
3E81 8E  3F38      04150 VERIFX LDX     #MESSG2
3E84 17  FEE6      04160        LBSR    MESSER
                   04170 *
3E87 17  FE9D      04180        LBSR    CONAIN
3E8A 17  FEB5      04190        LBSR    CONBCD
3E8D 17  FED0      04200        LBSR    ETYPE
3E90 A6  21        04210        LDA     1,Y
3E92 A7  06        04220        STA     6,X
                   04230 *
                   04240 * D POINTS TO THE START
                   04250 * ADDRESS IN THE EPROM,
```

```
                   04260 * Y POINTS TO THE DATA
                   04270 * STORED IN MEMORY.
                   04280 *
3E94 FC  3D01      04290        LDD     STTADR
3E97 10BE 3D03     04300        LDY     STTDAT
                   04310 *
                   04320 * A AND B ARE LATCHED AS
                   04330 * THE LSB AND MSB OF THE
                   04340 * EPROM'S ADDRESS, AND
                   04350 * ALL THE INFORMATION IN
                   04360 * PROGRESS IS DISPLAYED
                   04370 * FOR THE USER.
                   04380 *
3E9B E7  02        04390 AGIN2  STB     2,X
3E9D A7  04        04400        STA     4,X
3E9F 34  06        04410        PSHS    D
3EA1 17  FEEE      04420        LBSR    DISADR
3EA4 A6  84        04430        LDA     ,X
3EA6 34  20        04440        PSHS    Y
3EA8 108E 0478     04450        LDY     #$0478
3EAC 17  FEF2      04460        LBSR    SHOWAD
3EAF 86  7A        04470        LDA     #$7A
3EB1 A7  A0        04480        STA     ,Y+
3EB3 35  20        04490        PULS    Y
3EB5 A6  A4        04500        LDA     ,Y
3EB7 34  20        04510        PSHS    Y
3EB9 108E 047C     04520        LDY     #$047C
3EBD 17  FEE1      04530        LBSR    SHOWAD
3EC0 35  20        04540        PULS    Y
3EC2 A6  A4        04550        LDA     ,X
3EC4 A1  A0        04560        CMPA    ,Y+
3EC6 35  06        04570        PULS    D
3EC8 26  10        04580        BNE     VFAULT
3ECA C3  0001      04590 BACK   ADDD    #$0001
3ECD 10B3 3D05     04600        CMPD    ENDADR
3ED1 26  C8        04610        BNE     AGIN2
                   04620 *
                   04630 * EITHER A VERIFICATION
                   04640 * COMPLETE OR A VERIFY
                   04650 * FAILURE IS REPORTED IN
                   04660 * THE MESSAGES THAT
                   04670 * FOLLOW.
                   04680 *
3ED3 8E  3FA8      04690        LDX     #MESSG9
```

```
3ED6 17    FE94    04700  OUT1    LBSR    MESSER        3F09 A6    84      05150          LDA     ,X
3ED9 39             04710          RTS                  3F0B 34    22      05160          PSHS    A,Y
                    04720  *                            3F0D 108E  0478    05170          LDY     #$0478
3EDA 34    36       04730  VFAULT  PSHS    X,Y,D        3F11 17    FE8D    05180          LBSR    SHOWAD
3EDC 8E    3F58     04740          LDX     #MESSG4      3F14 35    22      05190          PULS    Y,A
3EDF 17    FE8B     04750          LBSR    MESSER       3F16 A7    A0      05200          STA     ,Y+
3EE2 35    36       04760          PULS    X,Y,D        3F18 35    06      05210          PULS    D
3EE4 20    E4       04770          BRA     BACK         3F1A C3    0001    05220          ADDD    #$0001
                    04780  *                            3F1D 10B3  3D05    05230          CMPD    ENDADR
                    04790  * THE ROUTINE TO READ THE    3F21 26    DD      05240          BNE     AGIN3
                    04800  * CONTENTS OF AN EPROM                05250  *
                    04810  * INTO MEMORY BEGINS HERE    3F23 8E    3FC8    05260          LDX     #MESSGB
                    04820  *                            3F26 20    AE      05270          BRA     OUT1
3EE6 8E    3FB8     04830  READRX  LDX     #MESSGA              05280  *
3EE9 17    FE81     04840          LBSR    MESSER               05290  *
                    04850  *                                    05300  * THE MESSAGES TO BE
                    04860  * THE USUAL SETUP OF PORT             05310  * DISPLAYED FOLLOW HERE.
                    04870  * A AS INPUT, PORTS B, C              05320  * NOT ALL THE MESSAGES
                    04880  * AND D AS OUTPUT FOLLOWS             05330  * ARE USED BY VERSION 1.0
                    04890  *                                     05340  * OF THE COLOR BURNER
3EEC 17    FE38     04900          LBSR    CONAIN               05350  * SOFTWARE.
3EEF 17    FE50     04910          LBSR    CONBCD               05360  *
3EF2 17    FE6B     04920          LBSR    ETYPE        3F28 50524F475241 05370 MESSG1   FCC    /PROGRAMMING/
3EF5 A6    21       04930          LDA     1,Y               4D4D494E47    (Note: All FCC output compressed -Ed)
3EF7 A7    06       04940          STA     6,X          3F33 6060       05380          FDB     $6060
                    04950  *                            3F35 6060       05390          FDB     $6060
                    04960  * D POINTS TO THE EPROM      3F37 60         05400          FCB     $60
                    04970  * START ADDRESS, Y POINTS     3F38 564552494659 05410 MESSG2  FCC    /VERIFYING/
                    04980  * TO THE STARTING PLACE            494E47
                    04990  * FOR DATA STORAGE IN        3F41 6060       05420          FDB     $6060
                    05000  * MEMORY.                    3F43 6060       05430          FDB     $6060
                    05010  *                            3F45 6060       05440          FDB     $6060
3EF9 FC    3D01     05020          LDD     STTADR       3F47 60         05450          FCB     $60
3EFC 10BE  3D03     05030          LDY     STTDAT       3F48 4552415345  05460 MESSG3   FCC    /ERASE/
                    05040  *                            3F4D 60         05470          FCB     $60
                    05050  * ADDRESSES ARE LATCHED      3F4E 434845434B  05480          FCC     /CHECK/
                    05060  * INTO THE EPROM, AND        3F53 6060       05490          FDB     $6060
                    05070  * THE INFORMATION IS         3F55 6060       05500          FDB     $6060
                    05080  * DISPLAYED FOR THE          3F57 60         05510          FCB     $60
                    05090  * USER.                      3F58 564552494659 05520 MESSG4  FCC    /VERIFY/
                    05100  *                            3F5E 60         05530          FCB     $60
3F00 E7    02       05110  AGIN3   STB     2,X          3F5F 4641494C53  05540          FCC     /FAILS/
3F02 A7    04       05120          STA     4,X          3F64 60         05550          FCB     $60
3F04 34    06       05130          PSHS    D            3F65 4154       05560          FCC     /AT/
3F06 17    FE89     05140          LBSR    DISADR       3F67 60         05570          FCB     $60
```

Circle No. 78 on Reader Service Card

```
3F68  4552415345  05580  MESSG5  FCC  /ERASE/
3F6D        60     05590          FCB  $60
3F6E  4641494C53  05600          FCC  /FAILS/
3F73        60     05610          FCB  $60
3F74      4154     05620          FCC  /AT/
3F76      6060     05630          FDB  $6060
3F78  4255524E    05640  MESSG6  FCC  /BURN/
3F7C        60     05650          FCB  $60
3F7D  4641494C53  05660          FCC  /FAILS/
3F82        60     05670          FCB  $60
3F83      4154     05680          FCC  /AT/
3F85      6060     05690          FDB  $6060
3F87        60     05700          FCB  $60
3F88  4255524E    05710  MESSG7  FCC  /BURN/
3F8C        60     05720          FCB  $60
3F8D  434F4D504C45 05730         FCC  /COMPLETE/
      5445
3F95      6060     05740          FDB  $6060
3F97        60     05750          FCB  $60
3F98  4550524F4D  05760  MESSG8  FCC  /EPROM/
3F9D        60     05770          FCB  $60
3F9E  455241534544 05780         FCC  /ERASED/
3FA4      6060     05790          FDB  $6060
3FA6      6060     05800          FDB  $6060
3FA8  564552494659 05810 MESSG9  FCC  /VERIFY/
3FAE        60     05820          FCB  $60
3FAF  434F4D504C45 05830         FCC  /COMPLETE/
      5445
3FB7        60     05840          FCB  $60
3FB8  52454144494E 05850 MESSGA  FCC  /READING/
      47
3FBF        60     05860          FCB  $60
3FC0  4550524F4D  05870          FCC  /EPROM/
3FC5      6060     05880          FDB  $6060
3FC7        60     05890          FCB  $60
3FC8  4550524F4D  05900  MESSGB  FCC  /EPROM/
3FCD        60     05910          FCB  $60
3FCE  52454144    05920          FCC  /READ/
3FD2      6060     05930          FDB  $6060
3FD4      6060     05940          FDB  $6060
3FD6      6060     05950          FDB  $6060
3FD8  424144      05960  MESSGC  FCC  /BAD/
3FDB        60     05970          FCB  $60
3FDC  4550524F4D  05980          FCC  /EPROM/
3FE1        60     05990          FCB  $60
3FE2  54595045    06000          FCC  /TYPE/
3FE6      6060     06010          FDB  $6060
3FE8  44424B495453 06020 MESSGD  FCC  /DBKITSZ/
      5A
3FEF        60     06030          FCB  $60
3FF0  524F58425552 06040         FCC  /ROXBURY/
      59
3FF7        60     06050          FCB  $60
                   06060  *            *
      3FF8         06070  ZZZZZZ  EQU  *
                   06080  *
      0000         06090          END
00000 TOTAL ERRORS
```

```
AGAIN   3DD1      MESSG9  3FA8      R2764    3D10
AGIN1   3E52      MESSGA  3FB8      READRX   3EE6
AGIN2   3E9B      MESSGB  3FC8      RSP1     3D18
AGIN3   3F00      MESSGC  3FD8      RSP2     3D1C
BACK    3ECA      MESSGD  3FE8      RSP3     3D20
CONAIN  3D27      MLOOP1  3D73      RSP4     3D24
CONAOT  3D34      O27128  3D13      S27128   3D15
CONBCD  3D42      O2716   3D07      S2716    3D09
CONFIG  3DC4      O2732   3D0B      S2732    3D0D
CONVRT  3DB4      O2764   3D0F      S2764    3D11
D50MS   3D85      OFFSET  3D00      SHOWAD   3DA1
DELAY1  3D7B      OSP1    3D17      SSP1     3D19
DISADR  3D92      OSP2    3D1B      SSP2     3D1D
EFAULT  3E7C      OSP3    3D1F      SSP3     3D21
ENDADR  3D05      OSP4    3D23      SSP4     3D25
ERASEX  3E3C      OUT1    3ED6      STTADR   3D01
ETYPE   3D60      P27128  3D16      STTDAT   3D03
LETTER  3DBB      P2716   3D0A      TAB128   3D13
LOOP1   3D7F      P2732   3D0E      TAB16    3D07
LOOP50  3D8A      P2764   3D12      TAB32    3D0B
MESSER  3D6D      PROGMX  3DBE      TAB64    3D0F
MESSG1  3F28      PROGRM  3DF1      TABSP1   3D17
MESSG2  3F38      PSP1    3D1A      TABSP2   3D1B
MESSG3  3F48      PSP2    3D1E      TABSP3   3D1F
MESSG4  3F58      PSP3    3D22      TABSP4   3D23
MESSG5  3F68      PSP4    3D26      VERIFX   3E81
MESSG6  3F78      R27128  3D14      VFAULT   3EDA
MESSG7  3F88      R2716   3D08      ZZZZZZ   3FF8
MESSG8  3F98      R2732   3D0C      ■■■
```

Dissecting

Your ROM

Extended Basic continues.
The tenth of 14 articles.

by Jake Commander

T HERE'S NOT TOO MUCH I can add this month — the code says it all by itself. Apart from the humble HEX$ function, the code splits nicely into two parts: that which deals with the Renumber (RENUM) command, and that which deals with RS-232-related input/output.

The Renumber routine operates in three passes. The first validates the line numbers to ensure all referenced line numbers are defined in the program. At the same time, all line numbers are con-

verted to a five-byte binary format. Then the start line numbers are renumbered, and finally the five-byte binary numbers are reconverted to ASCII and renumbered as appropriate. This can be a little difficult to follow, as a couple of subroutines operate differently on each pass.

Following on the heels of RENUM is the HEX$ function. This is another string function, which is only remarkable in that it's not included along with the string functions in last month's code.

The last block of code deals with the

DLOAD command, which hardly gets a mention in the version of the user's manual I'm blessed with. The mysterious protocol required to use the DLOAD command can be unravelled from this routine. Also included are a host of routines dealing with RS-232 interfacing, such as block and character I/O. These routines serve to make the Color Computer a much more useful machine than the manual alludes to. Why the DLOAD command is so downplayed still puzzles me.

■ ■ ■

```
Line   Code   Object
Numbr  Addr   Code        Mnemonic  Operand   Jake's Comment

01219  8A02   9E D1       LDX       <$D1      ;RENUM start line
01220  8A04   9F 2B       STX       <$2B      ;Save for search
01221  8A06   7E AD01     JMP       $AD01     ;Search for line number

RENUM new, start, inc
01222  8A09   BD AD26     JSR       $AD26     ;Reset variable pointers
01223  8A0C   CC 000A     LDD       #$000A    ;Default of 10
01224  8A0F   DD D5       STD       <$D5      ;Default new line number
01225  8A11   DD CF       STD       <$CF      ;Default increment
01226  8A13   5F          CLRB                ;Zero...
01227  8A14   DD D1       STD       <$D1      ;...default start line number
01228  8A16   9D A5       JSR       <$A5      ;Parse current chr
01229  8A18   24 06       BHS       $8A20     ;If non numeric
01230  8A1A   8D E0       BSR       $89FC     ;ASCII # to x
01231  8A1C   9F D5       STX       <$D5      ;Save new line number
01232  8A1E   9D A5       JSR       <$A5      ;Parse current chr
01233  8A20   27 1B       BEQ       $8A3D     ;If end of line
01234  8A22   BD B26D     JSR       $B26D     ;Syntax check for ","
01235  8A25   24 06       BHS       $8A2D     ;If non numeric chr after ","
01236  8A27   8D D3       BSR       $89FC     ;ASCII # to x
01237  8A29   9F D1       STX       <$D1      ;Save start line number
01238  8A2B   9D A5       JSR       <$A5      ;Parse current chr
01239  8A2D   27 0E       BEQ       $8A3D     ;If end of line
01240  8A2F   BD B26D     JSR       $B26D     ;Syntax check for ","
01241  8A32   24 06       BHS       $8A3A     ;If non numeric after ","
01242  8A34   8D C6       BSR       $89FC     ;ASCII # to x
01243  8A36   9F CF       STX       <$CF      ;Save increment
01244  8A38   27 49       BEQ       $8A83     ;If end of line
01245  8A3A   BD A5C7     JSR       $A5C7     ;?SN Error if not end of line
01246  8A3D   8D C3       BSR       $8A02     ;Search for start line number
01247  8A3F   9F D3       STX       <$D3      ;Save start line link pointer
01248  8A41   9E D5       LDX       <$D5      ;Search for new line number
01249  8A43   BC 8A04     CMPX      $8A04     ;New line link < start link?
01250  8A46   25 3A       BLO       $8A83     ;?FC Error if so
01251  8A48   8D 1C       BSR       $8A67     ;Get all new line numbers
01252  8A4A   BD 8ADD     JSR       $8ADD     ;Make line numbers binary
01253  8A4D   BD ACEF     JSR       $ACEF     ;Reset Basic link pointers
01254  8A50   8D AF       BSR       $8A02     ;Search for start line number
01255  8A52   9F D3       STX       <$D3      ;Save start line link pointer
01256  8A54   8D 3B       BSR       $8A91     ;Update line number gaps
01257  8A56   8D 36       BSR       $8A91     ;Replace all start line numbers
01258  8A58   8D 37       BSR       $8A91     ;Update line number gaps
01259  8A5A   BD 8B7B     JSR       $8B7B     ;Fill gaps with new line numbrs
01260  8A5D   BD AD26     JSR       $AD26     ;Reset variable pointers
01261  8A60   BD ACEF     JSR       $ACEF     ;Reset Basic link pointers
01262  8A63   7E AC73     JMP       $AC73     ;Goto direct mode

Get or replace start line numbers
01264  8A67   86 4F       LDA       #$4F      ;Flag 1st pass
-----  8A68   4F          CLRA                ;Flag 2nd pass
01265  8A69   97 D8       STA       <$D8      ;Save current pass number
01266  8A6B   9E D3       LDX       <$D3      ;Start line link pointer
01267  8A6D   0D D5       LDD       <$D5      ;New line number for it
01268  8A6F   8D 15       BSR       $8A86     ;RTS to caller if end prog
01269  8A71   0D D8       TST       <$D8      ;1st pass?
01270  8A73   26 02       BNE       $8A77     ;Yes, don't alter line numbers
01271  8A75   ED 02       STD       2,X       ;Else insert new line number

01272  8A77   AE 84       LDX       ,X        ;Get next line link
01273  8A79   8D 0B       BSR       $8A86     ;RTS to caller if end prog
01274  8A7B   D3 CF       ADDD      <$CF      ;Add inc to line number
01275  8A7D   25 04       BLO       $8A83     ;?FC Error if too big
01276  8A7F   81 FA       CMPA      #$FA      ;Line number LSB < 250?
01277  8A81   25 EE       BLO       $8A71     ;OK if so
01278  8A83   7E B44A     JMP       $B44A     ;?FC Error otherwise

RTS 1 back if end prog
01279  8A86   34 06       PSHS      A,B       ;Save new line number
01280  8A88   EC 84       LDD       ,X        ;End link?
01281  8A8A   35 06       PULS      A,B       ;Restore new line number
01282  8A8C   26 02       BNE       $8A90     ;If not end prog
01283  8A8E   32 62       LEAS      2,S       ;Else adj stak to skip nrml RTS
01284  8A90   39          RTS

Update line number gap info
01285  8A91   9E 19       LDX       <$19      ;Basic start address
01286  8A93   30 1F       LEAX      -1,X      ;Adjust for next op
01287  8A95   30 01       LEAX      1,X       ;=> next line link
01288  8A97   8D ED       BSR       $8A86     ;RTS to caller if end prog
01289  8A99   30 03       LEAX      3,X       ;Align before next chr
01290  8A9B   30 01       LEAX      1,X       ;Bump to next byte
01291  8A9D   A6 84       LDA       ,X        ;Get byte from Basic line
01292  8A9F   27 F4       BEQ       $8A95     ;If end of line
01293  8AA1   9F 0F       STX       <$0F      ;Hold line number gap start
01294  8AA3   4A          DECA                ;=1, at line # gap start?
01295  8AA4   27 0C       BEQ       $8AB2     ;Yes, search for ref'd line
01296  8AA6   4A          DECA                ;=2, at line # gap start?
01297  8AA7   27 2A       BEQ       $8AD3     ;If so
01298  8AA9   4A          DECA                ;=3, at line # gap start?
01299  8AAA   26 EF       BNE       $8A9B     ;No, get next byte
01300  8AAC   86 03       LDA       #$3       ;Flag for undefined line number
01301  8AAE   A7 80       STA       ,X+       ;Set flag & posn X
01302  8AB0   20 E7       BRA       $8A99     ;Skip over line number gap
01303  8AB2   EC 01       LDD       1,X       ;Line number MSB in A
01304  8AB4   6A 02       DEC       2,X       ;MSB really zero?
01305  8AB6   27 01       BEQ       $8AB9     ;No
01306  8AB8   4F          CLRA                ;Else reset MSB zero
01307  8AB9   E6 03       LDB       3,X       ;Line number LSB in B
01308  8ABB   6A 04       DEC       4,X       ;LSB really zero?
01309  8ABD   27 01       BEQ       $8AC0     ;No
01310  8ABF   5F          CLRB                ;Else reset LSB zero
01311  8AC0   ED 01       STD       1,X       ;Hold binary line number
01312  8AC2   DD 2B       STD.      <$2B      ;Save it for search
01313  8AC4   BD AD01     JSR       $AD01     ;Search for this line
01314  8AC7   9E 0F       LDX       <$0F      ;Restore line number gap start
01315  8AC9   25 E1       BLO       $8AAC     ;If line not found
01316  8ACB   6C 80       INC       ,X+       ;Set gap byte 0 to 1 or 2
01317  8ACD   ED 84       STD       ,X        ;Save referd line link or nmbr
01318  8ACF   20 C6       BRA       $8A99     ;Continue scan past this line
01319  8AD1   6F 84       CLR       ,X        ;To allow gap start = 1
01320  8AD3   AE 01       LDX       1,X       ;Get line pointer
01321  8AD5   AE 02       LDX       2,X       ;Get line number
01322  8AD7   9F 47       STX       <$47      ;Save for gap replacement
01323  8AD9   DC 47       LDD       <$47      ;Get line number
01324  8ADB   20 EA       BRA       $8AC7     ;Save & flag line number in gap
```

Make line numbers binary

Line Numbr	Code Addr	Object Code	Mnemonic	Operand	Jake's Comment
01325	8ADD	9E 19	LDX	<$19	;=> Basic start
01326	8ADF	20 04	BRA	$8AE5	;End-of-line pointer
01327	8AE1	9E A6	LDX	<$A6	;Bump to link pointer
01328	8AE3	30 01	LEAX	1,X	;RTS to caller if end prog
01329	8AE5	8D 9F	BSR	$8A86	;Bump over link pointer
01330	8AE7	30 02	LEAX	2,X	;Align before next chr
01331	8AE9	30 01	LEAX	1,X	;Parse next chr
01332	8AEB	9F A6	STX	<$A6	;Save as parse pointer
01333	8AED	9D 9F	JSR	<$9F	;Parse next chr
01334	8AEF	4D	TSTA		;End of line?
01335	8AF0	27 EF	BEQ	$8AE1	;Yes, align to next
01336	8AF2	2A F9	BPL	$8AED	;If not token, get next chr
01337	8AF4	9E A6	LDX	<$A6	;Parse pointer
01338	8AF6	81 FF	CMPA	#$FF	;Two-byte function?
01339	8AF8	27 EF	BEQ	$8AE9	;Yes, skip it
01340	8AFA	BD 01A0	JSR	$01A0	;RAM vector
01341	8AFD	81 A7	CMPA	#$A7	;"THEN"?
01342	8AFF	27 12	BEQ	$8B13	;Yes
01343	8B01	81 84	CMPA	#$84	;"ELSE"?
01344	8B03	27 0E	BEQ	$8B13	;Yes
01345	8B05	81 81	CMPA	#$81	;"GO"?
01346	8B07	26 E4	BNE	$8AED	;No, parse next chr
01347	8B09	9D 9F	JSR	<$9F	;Parse chr after "GO"
01348	8B0B	81 A5	CMPA	#$A5	;"TO"?
01349	8B0D	27 04	BEQ	$8B13	;If GOTO
01350	8B0F	81 A6	CMPA	#$A6	;"SUB"?
01351	8B11	26 D8	BNE	$8AEB	;If not GOSUB
01352	8B13	9D 9F	JSR	<$9F	;Parse possible line number
01353	8B15	25 04	BLO	$8B1B	;If ASCII 0-9
01354	8B17	9D A5	JSR	<$A5	;Else parse current chr again
01355	8B19	20 D4	BRA	$8AEF	;Cont scan of line
01356	8B1B	9E A6	LDX	<$A6	;Current parse pointer
01357	8B1D	34 10	PSHS	X	;Save target line # pointer
01358	8B1F	BD AF67	JSR	$AF67	;ASCII line # to integr in <$2B
01359	8B22	9E A6	LDX	<$A6	;Current parse pointer
01360	8B24	A6 82	LDA	,-X	;Get previous chr
01361	8B26	BD 90AA	JSR	$90AA	;Chr = ASCII 0-9?
01362	8B29	25 F9	BLO	$8B24	;Again if not
01363	8B2B	30 01	LEAX	1,X	;Bump past line number
01364	8B2D	1F 10	TFR	X,D	;Pointer LSB to B
01365	8B2F	E0 61	SUBB	1,S	;Get line number length
01366	8B31	C0 05	SUBB	#5	;Line number = 5 chrs?
01367	8B33	27 20	BEQ	$8B55	;No problem if so
01368	8B35	25 0A	BLO	$8B41	;If < 5 chr line number
01369	8B37	33 84	LEAU	,X	;Chr after line number = blank
01370	8B39	50	NEGB		;# chrs to make 5 chr gap
01371	8B3A	30 85	LEAX	B,X	;End of 5 chr gap = destination
01372	8B3C	BD 89B8	JSR	$89B8	;Copy down program
01373	8B3F	20 14	BRA	$8B55	;Now have 5 chr gap
01374	8B41	9F 47	STX	<$47	;1 past line number = copy end
01375	8B43	9E 1B	LDX	<$1B	;Program end pointer
01376	8B45	9F 43	STX	<$43	;= copy source
01377	8B47	50	NEGB		;# chrs required to open gap
01378	8B48	30 85	LEAX	B,X	;New program end pointer
01379	8B4A	9F 41	STX	<$41	;= copy destination
01380	8B4C	9F 1B	STX	<$1B	;Save new program end

Line Numbr	Code Addr	Object Code	Mnemonic	Operand	Jake's Comment
01381	8B4E	BD AC1E	JSR	$AC1E	;Copy up to open gap
01382	8B51	9E 45	LDX	<$45	;Last address copied
01383	8B53	9F A6	STX	<$A6	;New parse pointer
01384	8B55	35 10	PULS	X	;Target line number pointer
01385	8B57	86 01	LDA	#1	;Initial gap byte
01386	8B59	A7 84	STA	,X	;Byte 0 = fill
01387	8B5B	A7 02	STA	2,X	;MSB zero-flag
01388	8B5D	A7 04	STA	4,X	;LSB zero-flag
01389	8B5F	D6 2B	LDB	<$2B	;Line # MSB this point
01390	8B61	26 04	BNE	$8B67	;OK if non zero
01391	8B63	C6 01	LDB	#1	;Else must use non zero
01392	8B65	6C 02	INC	2,X	;Flag MSB really zero
01393	8B67	E7 01	STB	1,X	;Store MSB
01394	8B69	D6 2C	LDB	<$2C	;Line # LSB this point
01395	8B6B	26 04	BNE	$8B71	;OK if non zero
01396	8B6D	C6 01	LDB	#1	;Else must use non zero
01397	8B6F	6C 04	INC	4,X	;Flag LSB really zero
01398	8B71	E7 03	STB	3,X	;Store LSB
01399	8B73	9D A5	JSR	<$A5	;Parse current chr
01400	8B75	81 2C	CMPA	#$2C	;Comma?
01401	8B77	27 9A	BEQ	$8B13	;Yes, do line # list
01402	8B79	20 9C	BRA	$8B17	;Else cont line scan
01403	8B7B	9E 19	LDX	<$19	;=> Basic start
01404	8B7D	30 1F	LEAX	-1,X	;Adjust for next op
01405	8B7F	30 01	LEAX	1,X	;=> next line link
01406	8B81	EC 02	LDD	2,X	;Current line # as renumbered
01407	8B83	DD 68	STD	<$68	;Hold line #
01408	8B85	BD 8A86	JSR	$8A86	;RTS to caller if end prog
01409	8B88	30 03	LEAX	3,X	;Align before next chr
01410	8B8A	30 01	LEAX	1,X	;Bump to next byte
01411	8B8C	A6 84	LDA	,X	;Get byte from Basic line
01412	8B8E	27 EF	BEQ	$8B7F	;If end of line
01413	8B90	4A	DECA		;=1 at line gap start?
01414	8B91	27 1B	BEQ	$8BAE	;Yes
01415	8B93	80 02	SUBA	#2	;=3 at line gap start?
01416	8B95	26 F3	BNE	$8B8A	;No, get next byte
01417	8B97	34 10	PSHS	X	;Save gap pointer
01418	8B99	8E 8BD8	LDX	#$8BD8	;=>" UL "
01419	8B9C	BD B99C	JSR	$B99C	;Print message
01420	8B9F	AE E4	LDX	,S	;Restore gap pointer
01421	8BA1	EC 01	LDD	1,X	;Get undefined line pointer
01422	8BA3	BD BDCC	JSR	$BDCC	;Print undefined line #
01423	8BA6	BD BDC5	JSR	$BDC5	;Print "IN" & current line #
01424	8BA9	BD B958	JSR	$B958	;Print c/r
01425	8BAC	35 10	PULS	X	;Restore gap pointer
01426	8BAE	34 10	PSHS	X	;Save gap pointer
01427	8BB0	EC 01	LDD	1,X	;Get binary line # from gap
01428	8BB2	DD 52	STD	<$52	;Save in FPAC1
01429	8BB4	BD 880E	JSR	$880E	;Convert to floating point
01430	8BB7	BD BDD9	JSR	$BDD9	;Convert FPAC1 to ASCII
01431	8BBA	35 40	PULS	U	;Restore gap pointer
01432	8BBC	C6 05	LDB	#5	;# line + chrs
01433	8BBE	30 01	LEAX	1,X	;=> next chr in ASCII buffer
01434	8BC0	A6 84	LDA	,X	;Get ASCII chr
01435	8BC2	27 05	BEQ	$8BC9	;If end of string
01436	8BC4	5A	DECB		;Keep tally of chrs filled
01437	8BC5	A7 C0	STA	,U+	;Place ASCII # in gap

Line Numbr	Code Addr	Object Code	Mnemonic	Operand	Jake's Comment
01438	8BC7	20 F5	BRA	$8BBE	;Do next ASCII #
01439	8BC9	30 C4	LEAX	,U	;Current Basic mid line pointer
01440	8BCB	5D	TSTB		;Did 5 chr fill?
01441	8BCC	27 BE	BEQ	$8B8C	;Yes, continue line scan
01442	8BCE	31 C4	LEAY	,U	;Hold next byte pointer
01443	8BD0	33 C5	LEAU	B,U	;Get source to fill gap from
01444	8BD2	BD 89B8	JSR	$89B8	;Copy to crnt posn til progend
01445	8BD5	30 A4	LEAX	,Y	;Restore next byte pointer
01446	8BD7	20 B3	BRA	$8B8C	;Continue line scan
01447	8BD9	55			;U
01448	8BDA	4C	INCA		;L
01449	8BDB	20 00	BRA	$8BDD	;Space/end
HEX$					
01450	8BDD	BD B740	JSR	$B740	;Get # < 65536 to FPAC1
01451	8BE0	8E 03D9	LDX	#$03D9	;=> HEX$ string destination
01452	8BE3	C6 04	LDB	#4	;# hex digits to do = 4
01453	8BE5	34 04	PSHS	B	;Save # characters of hex
01454	8BE7	5F	CLRB		;Initialize hex digit
01455	8BE8	86 04	LDA	#4	;# bits per digit
01456	8BEA	08 53	ASL	<$53	;LSB left
01457	8BEC	09 52	ROL	<$52	;MSB left
01458	8BEE	59	ROLB		;Into B
01459	8BEF	4A	DECA		;Done 4 bits?
01460	8BF0	26 F8	BNE	$8BEA	;Cont until done
01461	8BF2	5D	TSTB		;Any # in B?
01462	8BF3	26 0A	BNE	$8BFF	;No, do ASCII
01463	8BF5	A6 E4	LDA	,S	;# characters left to do
01464	8BF7	4A	DECA		;This last character?
01465	8BF8	27 05	BEQ	$8BFF	;Yes, place in string
01466	8BFA	8C 03D9	CMPX	#$03D9	;At start of string?
01467	8BFD	27 0C	BEQ	$8C0B	;Yes, suppress leading zeroes
01468	8BFF	CB 30	ADDB	#$30	;Convert # to ASCII
01469	8C01	C1 39	CMPB	#$39	;>9?
01470	8C03	23 02	BLS	$8C07	;No, # is OK
01471	8C05	CB 07	ADDB	#7	;Else convert to A-F
01472	8C07	E7 80	STB	,X+	;Into string & bump
01473	8C09	6F 84	CLR	,X	;Delimit string with zero
01474	8C0B	35 04	PULS	B	;# chrs left to do
01475	8C0D	5A	DECB		;Done all?
01476	8C0E	26 D5	BNE	$8BE5	;No, do next
01477	8C10	32 62	LEAS	2,S	;Pull return
01478	8C12	8E 03D8	LDX	#$03D8	;=> string-1
01479	8C15	7E B518	JMP	$B518	;String info to string stack
DLOAD					
01480	8C18	BD A429	JSR	$A429	;Close files
01481	8C1B	6F E2	CLR	,-S	;Init 'Basic' load flag
01482	8C1D	81 4D	CMPA	#$4D	;"M"?
01483	8C1F	26 04	BNE	$8C25	;No
01484	8C21	A7 E4	STA	,S	;Save "M" load flag
01485	8C23	9D 9F	JSR	<$9F	;Parse next chr
01486	8C25	BD A578	JSR	$A578	;Initialize filename
01487	8C28	9D A5	JSR	<$A5	;Parse current chr
01488	8C2A	27 18	BEQ	$8C44	;If EOL, deflt to last baud set
01489	8C2C	BD B26D	JSR	$B26D	;Syntax check for " "
01490	8C2F	81 2C	CMPA	#$2C	;","?
01491	8C31	27 11	BEQ	$8C44	;if comma
01492	8C33	BD B70B	JSR	$B70B	;Compute # < 256 to B
01493	8C36	86 B0	LDA	#$B0	;Baud rate delay for 300 baud
01494	8C38	5D	TSTB		;Baud rate 0 (300)
01495	8C39	27 07	BEQ	$8C42	;Yes, 300 baud
01496	8C3B	86 2C	LDA	#$2C	;Baud rate delay=1200
01497	8C3D	5A	DECB		;Baud rate 1 (1200)?
01498	8C3E	1026 2808	LBNE	$B44A	;?FC Error if baud not 0 or 1
01499	8C42	97 E6	STA	<$E6	;Setup baud rate delay
01500	8C44	BD 8CD0	JSR	$8CD0	;Get lst I/P block to buffer
01501	8C47	34 02	PSHS	A	
01502	8C49	86 FD	LDA	#$FD	;-3
01503	8C4B	97 6F	STA	<$6F	;Set device # = -3
01504	8C4D	35 02	PULS	A	
01505	8C4F	6D E0	TST	,S+	;DLOAD"M"?
01506	8C51	26 32	BNE	$8C85	;If so
01507	8C53	BD A5C7	JSR	$A5C7	;?SN Error if not EOL
01508	8C56	5D	TSTB		;ASCII file?
01509	8C57	27 06	BEQ	$8C5F	;No, FM Error
01510	8C59	BD AD19	JSR	$AD19	;Initialize Basic cold
01511	8C5C	7E AC7C	JMP	$AC7C	;To direct mode
01512	8C5F	7E A616	JMP	$A616	;?FM Error
Extended CLOADM					
01513	8C62	9D 9F	JSR	<$9F	;Parse next chr
01514	8C64	81 4D	CMPA	#$4D	;"M"?
01515	8C66	1026 182E	LBNE	$A498	;Normal CLOAD if not
01516	8C6A	0F 78	CLR	<$78	;File mode = header
01517	8C6C	9D 9F	JSR	<$9F	;Parse next chr
01518	8C6E	BD A578	JSR	$A578	;Initialize filename
01519	8C71	BD A648	JSR	$A648	;Search for cassette file
01520	8C74	7D 01E4	TST	$01E4	;Normal CLOADM
01521	8C77	1027 188A	LBEQ	$A505	;File type
01522	8C7B	FE 01E2	LDU	$01E2	
01523	8C7E	0A 6F	DEC	<$6F	;Set device number = -1
01524	8C80	BD A635	JSR	$A635	;Read block to mem. Refresh ptr
01525	8C83	1F 30	TFR	U,D	
01526	8C85	83 0200	SUBD	#$0200	;File type = binary?
01527	8C88	26 D5	BNE	$8C5F	;?FM Error if not
01528	8C8A	9E 8A	LDX	<$8A	;Zero constant default offset
01529	8C8C	9D A5	JSR	<$A5	;Parse current chr
01530	8C8E	27 06	BEQ	$8C96	;If end of line
01531	8C90	BD B26D	JSR	$B26D	;Syntax check for " "
01532	8C93	BD B73D	JSR	$B73D	;Get # < 65536 to X
01533	8C96	9F D3	STX	<$D3	;Offset
01534	8C98	BD A5C7	JSR	$A5C7	;Syntax check end of statement
01535	8C9B	8D 29	BSR	$8CC6	;Get I/P chr from buffer
01536	8C9D	34 02	PSHS	A	;EOF marker
01537	8C9F	8D 1E	BSR	$8CBF	;Input 2-byte address to D
01538	8CA1	1F 02	TFR	D,Y	;# bytes to Y
01539	8CA3	8D 1A	BSR	$8CBF	;Input 2-byte address to D
01540	8CA5	D3 D3	ADDD	<$D3	;Add offset
01541	8CA7	DD 9D	STD	<$9D	;Load address...
01542	8CA9	1F 01	TFR	D,X	;...to X
01543	8CAB	A6 E0	LDA	,S+	;EOF?
01544	8CAD	1026 177C	LBNE	$A42D	;Close if so
01545	8CB1	8D 13	BSR	$8CC6	;Get I/P chr from buffer
01546	8CB3	A7 84	STA	,X	;Store in mem
01547	8CB5	A1 80	CMPA	,X+	;Went in OK?

PETROCCI FREELANCE ASSOCIATES

FAMILY

A genealogical data base for the Color Computer

Maintains data on 255 people in first eight generations of your family tree. Prints 3 charts; 5 generation pedigree - graphic display of lineage; Family group charts; ancestors by reference number. Easy to use.
32K EXT 14.95

ALPHABET SONG

An excellent companion to preschool package.

What better way to teach the alphabet than with a familiar tune accompanied by full size screen display of the letters. This one has been kid tested and loved!

Level 1 - sequences through entire alphabet with song.
Level 2 - -steps at random letters, child must press next letter in sequence
Level 3 - Child must enter each letter in order.
All levels accompanied by Alphabet Song.
16K EXT 11.95

See Special
Prices Below

SPRING SPECIAL
ALL GAMES REDUCED 30%

Good Through
June 30, 1984

- **INSPECTOR CLUESEAU**
  The No. 1 Murder Mystery for the CoCo!
  (Every game is different.) 32K EXT 13.95
- **PATTI-PAK**
  "The Best" - Highest Quality Graphics
  16K 15.95
- **STAGECOACH**
  Wild West Adventure! 16K EXT 13.95

- **FLASHER**
  Can Your Memory Save You From The Sharks?
  32K EXT 13.95
- **TYCOON**
  CoCo Monopoly. 32K Tape Only 13.95
- **BOWLER**
  Sports Action Fun! 16K EXT 13.95
- **DUNKA DUCK**
  Arcade Action For All Ages. 16K EXT 13.95

BOWLING SECRETARY
(New Super Second Edition)

Now includes handicap routines for both men and women, pin spotting, selection of up to 15 players per team, plus the standard team standings, individual average, high and total pins, team won/lost, high series, cumulative total team points.

Printer Output & Screen Display
32K EXT - 24.95   Std. 16K Version Still Available

LITTLE LEAGUER

Does all the record keeping for you. Excellent printout. Allows mid season entry. Keeps 19 different statistics and individual as well as team statistics.

Easy screen editing features - allows editing of single player data. Saves to tape or disk.
32K EXT 24.95

Statistical Analysis
32K EXT 2 Programs for 34.95
Stress Evaluator
16K EXT 24.95
Medical Terminology
32K EXT 19.95
Heart Lung Circulatory
32K EXT 34.95
Weather Forecaster
32K EXT DISK ... 19.95
Weather Watch
16K EXT 24.95
Hurricane Tracker
16K EXT 15.95
Print Spooler
64K 11.95

Master Graphics Tool Kit
32K EXT 39.95

Text Master Graphics
32K EXT 19.95

All Programs 16K Tape
Unless Otherwise Specified
All Programs Available on Disk - Add $5.00
Special Sale Prices - Retail Only

Include $1.50 for handling for each program.
Arizona residents add 7% sales tax.
Quantity Discounts to Dealers

PETROCCI FREELANCE ASSOCIATES
651 N. Houghton Rd
Tucson, AZ 85748
602-296-1041

Super Disk Utility
32K EXT DISK 44.95
Real Estate Investment
16K EXT 24.95
Homeowner Selling Analysis
16K EXT 24.95
Owner Financed Real Estate
16K EXT 24.95
KIDS KORNER
Preschool Package, ABCs,
123s, Shapes, Big-Bigger
All Four 24.95
Guillotine Spelling
Game 16K EXT 9.95
Alphabet Song
16K EXT 11.95

```
Line    Code    Object
Numbr   Addr    Code      Mnemonic  Operand        Jake's Comment

01548   8CB7    26 14     BNE       $8CCD          ;?IO Error if not
01549   8CB9    31 3F     LEAY      -1,Y           ;Done all?
01550   8CBB    26 F4     BNE       $8CB1          ;Continue if not
01551   8CBD    20 DC     BRA       $8C9B          ;Do next block

Input 2-byte address to D
01552   8CBF    8D 00     BSR       $8CC1          ;Get #1 chr to A, #2 chr to B
01553   8CC1    8D 03     BSR       $8CC6          ;Get I/P chr from buffer
01554   8CC3    1E 89     EXG       A,B
01555   8CC5    39        RTS

01556   8CC6    BD A176   JSR       $A176          ;Get I/P chr
01557   8CC9    0D 70     TST       <$70           ;Buffer empty?
01558   8CCB    27 F8     BEQ       $8CC5          ;RTS if not
01559   8CCD    7E A619   JMP       $A619          ;Else ?IO Error

01560   8CD0    8D 42     BSR       $8D14          ;O/P filename, get file type
01561   8CD2    34 06     PSHS      A,B            ;Type & mode
01562   8CD4    4C        INCA                     ;?NE Error if so
01563   8CD5    27 06     BEQ       $8CDD          ;=$FF?
01564   8CD7    DE 8A     LDU       <$8A           ;Zero constant
01565   8CD9    8D 09     BSR       $8CE4          ;Read next RS232 block
01566   8CDB    35 86     PULS      A,B,PC         ;Restore type & mode
01567   8CDD    C6 34     LDB       #$34           ;Error code
01568   8CDF    7E AC46   JMP       $AC46          ;Error processing

Read next RS232 block
01569   8CE2    DE 7E     LDU       <$7E           ;= I/O address
01570   8CE4    30 41     LEAX      1,U            ;Add one
01571   8CE6    9F 7E     STX       <$7E           ;Restore I/O address
01572   8CE8    8E 01DA   LDX       #$01DA         ;=> input buffer
01573   8CEB    8D 8D7C   JSR       $8D7C          ;Read RS232 block to buffer
01574   8CEE    7E A644   JMP       $A644          ;Setup # bytes,& refrsh I/P ptr

Called during character I/P from $A176
01575   8CF1    96 6F     LDA       <$6F           ;Device number
01576   8CF3    81 FD     CMPA      #$FD           ;-32
01577   8CF5    26 0A     BNE       $8D01          ;RTS if not
01578   8CF7    32 62     LEAS      2,S            ;Pull return
01579   8CF9    0F 70     CLR       <$70           ;Set I/O buffer unflushed
01580   8CFB    0D 79     TST       <$79           ;More chrs to extrct frm buffr?
01581   8CFD    26 03     BNE       $8D02          ;Yes
01582   8CFF    03 70     COM       <$70           ;Else flag buffer = empty
01583   8D01    39        RTS

Get RS232 character from buffer
01584   8D02    34 74     PSHS      B,X,Y,U
01585   8D04    9E 7A     LDX       <$7A           ;I/O buffer pointer
01586   8D06    A6 80     LDA       ,X+            ;Get chr from buffer, bump ptr
01587   8D08    34 02     PSHS      A              ;Hold chr
01588   8D0A    9F 7A     STX       <$7A           ;Save new ptr
01589   8D0C    0A 79     DEC       <$79           ;1 less chr in buffer
01590   8D0E    26 02     BNE       $8D12          ;If buffer has more
01591   8D10    8D D0     BSR       $8CE2          ;Read next RS232 I/P block
01592   8D12    35 F6     PULS      A,B,X,Y,U,PC

Output filename, get file type
01593   8D14    4F        CLRA                     ;Initialize # errors

01594   8D15    34 16     PSHS      A,B,X          ;# errs, chksm, spc for I/P adr
01595   8D17    31 E4     LEAY      ,S             ;Hold stack pointer in Y
01596   8D19    20 02     BRA       $8D1D

01597   8D1B    8D 2B     BSR       $8D48          ;Check no more than 5 errors
01598   8D1D    86 8A     LDA       #$8A           ;Required I/P byte
01599   8D1F    8D 37     BSR       $8D58          ;O/P then I/P specified byte
01600   8D21    26 F8     BNE       $8D1B          ;If bad byte or I/O
01601   8D23    8E 01D2   LDX       #$01D2         ;=> filename to O/P
01602   8D26    A6 80     LDA       ,X+            ;Filename character
01603   8D28    BD 8E04   JSR       $8E04          ;O/P RS232 chr with chksm
01604   8D2B    8C 01DA   CMPX      #$01DA         ;Done all filename?
01605   8D2E    26 F6     BNE       $8D26          ;If not
01606   8D30    8D 30     BSR       $8D62          ;O/P chksm then I/P byte = $C8
01607   8D32    26 E7     BNE       $8D1B          ;If bad
01608   8D34    8D 3C     BSR       $8D72          ;Get RS232 I/P byte with chksm
01609   8D36    26 E3     BNE       $8D1B          ;I/O error
01610   8D38    A7 22     STA       2,Y            ;Save file type
01611   8D3A    8D 36     BSR       $8D72          ;Get RS232 I/P byte with chksm
01612   8D3C    26 DD     BNE       $8D1B          ;I/O error
01613   8D3E    A7 23     STA       3,Y            ;Save file mode
01614   8D40    8D 29     BSR       $8D6B          ;Get checksum
01615   8D42    26 D7     BNE       $8D1B          ;If bad checksum or I/O
01616   8D44    32 62     LEAS      2,S            ;Pull # errors + checksum
01617   8D46    35 86     PULS      A,B,PC         ;Pull type & mode

Check no more than 5 errors
01618   8D48    6C A4     INC       ,Y             ;Bump # errors
01619   8D4A    A6 A4     LDA       ,Y             ;Get current # errors
01620   8D4C    81 05     CMPA      #$5            ;Less than 5?
01621   8D4E    25 1A     BLO       $8D6A          ;RTS if less than 5
01622   8D50    86 BC     LDA       #$BC           ;Byte to O/P
01623   8D52    BD 8E0C   JSR       $8E0C          ;O/P RS232 byte
01624   8D55    7E A619   JMP       $A619          ;?IO Error

Output then input specified byte
01625   8D58    34 02     PSHS      A              ;Save required input byte
01626   8D5A    8D 5C     BSR       $8DB8          ;Clear chksm, o/p then i/p byte
01627   8D5C    26 02     BNE       $8D60          ;If bad I/O
01628   8D5E    A1 E4     CMPA      ,S             ;This the byte required?
01629   8D60    35 82     PULS      A,PC           ;Exit with I/O or cmpare status

Output checksum then input $C8
01630   8D62    A6 21     LDA       1,Y            ;Checksum
01631   8D64    8D 52     BSR       $8DB8          ;Clear chksm, o/p then i/p byte
01632   8D66    26 02     BNE       $8D6A          ;RTS if bad I/O
01633   8D68    81 C8     CMPA      #$C8           ;Set flags as per compare
01634   8D6A    39        RTS

Get RS232 checksum byte
01635   8D6B    8D 05     BSR       $8D72          ;Get RS232 input byte
01636   8D6D    26 FB     BNE       $8D6A          ;RTS if bad I/O
01637   8D6F    A6 21     LDA       1,Y            ;Checksum should be zero
01638   8D71    39        RTS

Get RS232 byte, do checksum
01639   8D72    8D 48     BSR       $8DBC          ;Input RS232 byte
01640   8D74    34 03     PSHS      CCR,A          ;Save status + input byte
```

```
Line  Code  Object
Numbr Addr  Code    Mnemonic Operand     Jake's Comment

01641 8D76 A8 21    EORA 1,Y             ;Checksum with total
01642 8D78 A7 21    STA  1,Y             ;New checksum
01643 8D7A 35 83    PULS CCR,A,PC

Read RS232 block to buffer
01644 8D7C 4F       CLRA                 ;Init # errors
01645 8D7D 34 76    PSHS A,B,X,Y,U       ;#errs, cksm,=>i/p buf,blklen,0
01646 8D7F 68 67    ASL  6,S
01647 8D81 69 66    ROL  5,S
01648 8D83 64 67    LSR  7,S             ;Hold currnt stack pointer in Y
01649 8D85 31 E4    LEAY ,S
01650 8D87 20 02    BRA  $8D8B
01651 8D89 8D BD    BSR  $8D48           ;Check no more than 5 errors
01652 8D8B 86 97    LDA  #$97            ;Required byte
01653 8D8D 8D C9    BSR  $8D58           ;O/P then I/P specified byte
01654 8D8F 26 F8    BNE  $8D89           ;If bad byte or I/O
01655 8D91 A6 26    LDA  6,Y             ;O/P RS232 byte with checksum
01656 8D93 8D 6F    BSR  $8E04
01657 8D95 A6 27    LDA  7,Y
01658 8D97 8D 6B    BSR  $8E04           ;O/P RS232 byte with checksum
01659 8D99 8D C7    BSR  $8D62           ;O/P checksum then I/P $C8
01660 8D9B 26 EC    BNE  $8D89           ;If I/O error or bad byte
01661 8D9D 8D 19    BSR  $8DB8           ;Get RS232 byte with checksum
01662 8D9F 26 E8    BNE  $8D89           ;If I/O error
01663 8DA1 A7 24    STA  4,Y             ;= # bytes in block
01664 8DA3 AE 22    LDX  2,Y             ;=> input buffer
01665 8DA5 C6 80    LDB  #$80            ;= 128 bytes to input
01666 8DA7 8D 0F    BSR  $8DB8           ;Get RS232 byte with checksum
01667 8DA9 26 DE    BNE  $8D89           ;If I/O error
01668 8DAB A7 80    STA  ,X+             ;Byte to buffer & bump ptr
01669 8DAD 5A       DECB                 ;Done whole block?
01670 8DAE 26 F7    BNE  $8DA7           ;Continue if not
01671 8DB0 8D B9    BSR  $8D6B           ;Get RS232 checksum byte
01672 8DB2 26 D5    BNE  $8D89           ;If I/O or checksum error
01673 8DB4 32 64    LEAS 4,S             ;Pull #errs, chksm,=> I/P buffr
01674 8DB6 35 96    PULS A,B,X,PC        ;A=blklen

Input RS232 byte to A
01675 8DB8 6F 21    CLR  1,Y             ;Clear checksum
01676 8DBA 8D 50    BSR  $8E0C           ;O/P RS232 byte first
01677 8DBC 4F       CLRA                 ;Initialize I/O flag = OK
01678 8DBD 34 15    PSHS CCR,B,X         ;Disable interrupts
01679 8DBF 1A 50    ORCC #$50
01680 8DC1 96 E7    LDA  <$E7            ;Timeout counter
01681 8DC3 9E 8A    LDX  <$8A            ;Zero constant for timeout
01682 8DC5 8D 1F    BSR  $8DE6           ;Get RS232 input level
01683 8DC7 24 FC    BHS  $8DC5           ;Again if 0 level
01684 8DC9 8D 1B    BSR  $8DE6           ;Get RS232 input level
01685 8DCB 25 FC    BLO  $8DC9           ;Again if 1 level
01686 8DCD 8D 2A    BSR  $8DF9           ;Baud delay
01687 8DCF C6 01    LDB  #1              ;I/P scan bit ptr
01688 8DD1 34 04    PSHS B               ;Save current scan bit
01689 8DD3 4F       CLRA                 ;Initialize input
01690 8DD4 8D 21    BSR  $8DF7           ;Baud delay times 2
01691 8DD6 F6 FF22  LDB  $FF22           ;Get RS232 input bit...

01692 8DD9 56       RORB                 ;...into carry flag
01693 8DDA 24 02    BHS  $8DDE           ;If zero
01694 8DDC AA E4    ORA  ,S              ;Else current scan bit = on
01695 8DDE 68 E4    ASL  ,S              ;Next scan bit left
01696 8DE0 24 F2    BHS  $8DD4           ;No crry flg if not done 8 bits
01697 8DE2 32 61    LEAS 1,S             ;Pull scan bit ptr
01698 8DE4 35 95    PULS CCR,B,X,PC

Get RS232 input level
01699 8DE6 F6 FF22  LDB  $FF22           ;=> RS232 input in bit 0
01700 8DE9 56       RORB                 ;Received bit to carry flag
01701 8DEA 30 01    LEAX 1,X             ;Bump timeout sub-count
01702 8DEC 26 08    BNE  $8DF6           ;RTS if OK
01703 8DEE 4A       DECA                 ;Decrement main timeout count
01704 8DEF 26 05    BNE  $8DF6           ;RTS if still time
01705 8DF1 32 62    LEAS 2,S             ;Pull return
01706 8DF3 35 15    PULS CCR,B,X         ;Align stack
01707 8DF5 4C       INCA                 ;Set I/O flag = bad
01708 8DF6 39       RTS

Baud rate delay
01709 8DF7 8D 00    BSR  $8DF9           ;Entry point for delay*2
01710 8DF9 34 02    PSHS A               ;Get baud rate delay
01711 8DFB 96 E6    LDA  <$E6
01712 8DFD 21 FE    BRN  $8DFD           ;NOP
01713 8DFF 4A       DECA
01714 8E00 26 FB    BNE  $8DFD           ;Finished?
01715 8E02 35 82    PULS A,PC            ;Go again if not

Output RS232 byte
01716 8E04 34 02    PSHS A               ;E.P. to O/P byte & form chksm
01717 8E06 A8 21    EORA 1,Y             ;Update checksum
01718 8E08 A7 21    STA  1,Y             ;Restore it
01719 8E0A 35 02    PULS A               ;Realign stack
01720 8E0C 34 07    PSHS CCR,A,B         ;Disable interrupts
01721 8E0E 1A 50    ORCC #$50            ;Baud delay*2
01722 8E10 8D E5    BSR  $8DF7           ;Baud delay*2
01723 8E12 8D E3    BSR  $8DF7           ;Drop voltge level for strt bit
01724 8E14 7F FF20  CLR  $FF20           ;Baud delay*2
01725 8E17 8D DE    BSR  $8DF7           ;Initialize B to point at bit 0
01726 8E19 C6 01    LDB  #1              ;Save bit pointer
01727 8E1B 34 04    PSHS B               ;Byte to O/P
01728 8E1D A6 62    LDA  2,S             ;Current bit=1?
01729 8E1F A4 E4    ANDA ,S              ;No O/P zero bit
01730 8E21 27 02    BEQ  $8E25           ;Current bit=1?
01731 8E23 86 02    LDA  #2              ;Else O/P a one bit
01732 8E25 B7 FF20  STA  $FF20           ;RS232 data O/P
01733 8E28 8D CD    BSR  $8DF7           ;Baud delay*2
01734 8E2A 68 E4    ASL  ,S              ;Scan next bit to the left
01735 8E2C 24 EF    BHS  $8E1D           ;If not done 8 bits
01736 8E2E 86 02    LDA  #2              ;Raise level...
01737 8E30 B7 FF20  STA  $FF20           ;...for mark
01738 8E33 32 61    LEAS 1,S             ;Pull bit pointer
01739 8E35 35 87    PULS CCR,A,B,PC
```

Circle No. 37 on Reader Service Card

The Devil's Disk Editor

by William Barden, Jr.

16K Disk Extended Color Basic

I WON'T GO INTO A LENGTHY discussion of the program here, but will give you a thumbnail sketch of how it works.

The file dump portion of the program assembles the file name from the user input and then reads in the nine sectors of the directory from disk, one sector at a time. A search is made of each directory sector to find the file named. If the file name is found, the file location is picked up from the fourteenth byte of the directory entry. The FAT (File Allocation Table) is then read into memory as string variable P$. The first sector of the file is then read and displayed.

The sector dump portion of the program assembles the drive number, track number, and sector number and then reads and displays the desired sector.

The GOTO next block portion of the program increments the offset by 64 bytes, reads in the next sector if necessary (every fourth display), and displays the next block. If a named file is involved, the process is similar, but the next granule is found if necessary.

The GOTO previous block portion of the program decrements the offset by 64 bytes, reads in the previous sector if necessary (every fourth display), and displays the previous block. If a named file is involved, the process is similar, but the previous granule is found if necessary.

The modify memory mode portion of the program uses variables CL (current line) and CC (current character position) to display the cursor. Checks are made of digits entered to make certain they are valid hexadecimal digits. Pushing the Enter key causes a write to the current

sector after the data is assembled from the screen display.

The read in and display current sector subroutine uses DSKI$ to read in two strings of 128 bytes each, representing the current sector. 64 bytes of one of the strings is displayed, depending upon the current offset.

The list of variables and their use follows:

A$ = temporary string variable
A = temporary variable
AC = character in modify mode
AL = left character in modify mode
AR = right character in modify mode
C = column in modify mode, others
CC = current character position — used in modify mode
CL = current line — used in modify mode
CM = temporary variable — holds CP in modify mode
CP = cursor position — used in modify mode
D = disk drive number, 0 to 3
F$ = file name for named file
F1$ = name of file in start file
F2$ = extension of file in start file
FT = holds FAT index
I = temporary variable
J = temporary variable
LS = − 1 if no current sector
NS = number of sectors in current gran-

ule of named file — 9 if not last granule, 0 – 9 if last granule
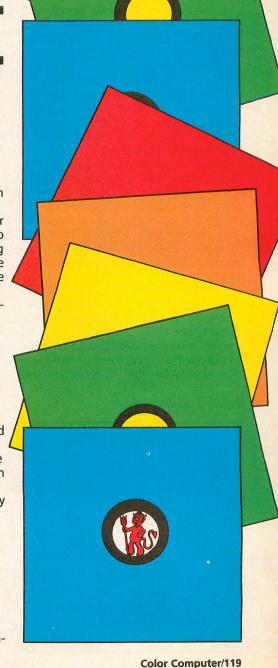
O = offset, 0 to 3

OF = original FAT index

P$ = portion of FAT for named file

R = row in modify, others

RN = *record number* of named file

S$ = current directory sector in start file

S0$ – S3$ = hold 64 bytes each of current sector read in

S = sector number, 1 to 18

SH$ = holds first 128 bytes in modify, others

SL$ = holds last 128 bytes in modify, others

T = track number, 0 to 34

About That Killed File

I have a description of file formats and also hints and kinks about using Devil's Disk Editor in its manual, but let's tackle the problem of killed files here. How to restore a killed file?

The Disk Operating System kills a file by "zeroing out" the first byte of the directory entry, as shown in Figure 1. It also deallocates the FAT entries, changing them from links to $FF. The $FF indicates that the granule is unused.

Before doing anything, back up the disk containing the killed file and use the back-up disk for the following operations. By the way, the killed file is still intact only if you haven't done any other disk writes after killing the file. If you have done disk writes after the kill, all or a portion of the file may have been overwritten!

To restore a killed file, you must first find every granule that contains the file. This must be done by searching the disk sectors, but here's a hint of where to begin: Using the Devil's Disk Editor, look in the directory entry for your killed file. You should be able to recognize the

name. Change the first character of the name back to the original, making certain that the proper name is shown in the ASCII portion of the display. Rewrite the directory sector with an Enter after the modify mode.

Now find the "location" value from the fourteenth byte of the directory entry. Divide this value by two. The quotient is the track number and the remainder of zero or one defines the granule in the track, either the track starting at

---

"Remember that the Devil's Disk Editor is a two-edged utility."

---

sector one or the granule starting at sector ten. Add one to the track number if the track is above track 16 to compensate for the two directory granules.

Now display the granule in the track and sector you've found. There's a good chance the granule may hold the entire file if the file is a short Basic program or other short file. Use a listing of your program for comparison. If you're looking for a Basic program, remember that the Basic commands will be "tokenized" to one-byte form, so you won't be able to recognize the commands. (See my IJG book, *How to Do It On the TRS-80*, for more information on tokens.)

If the first granule does not contain the entire program, you'll have to continue the search in other granules. Jot down the track, sector number of the first granule and then look at all FAT entries (track

17, sector 2) that contain an $FF. Divide by two as before to find the corresponding track and sector number, remembering that values for tracks should have one added if the track is greater than 16. Remember that granules start on sector one or ten of each track; you should be able to investigate only the first block of the first sector of a granule to see if the data there looks like your killed file. Jot down every granule that is used by track and sector number.

When you have found every granule that was previously used in your file, you have one more chore: Look at the last granule and find the last sector used to hold your file. You should be able to do this by comparison with a Basic or other listing. You'll have from one to nine sectors that are used in this last granule.

You'll now have a list similar to this:

track 20, sector 1
track 22, sector 10
track 27, sector 1, 5 sectors used

The first granule used should already be in the directory entry, so you'll be working only with FAT entries. Read in the FAT. Now convert the track, sector number to granule numbers by subtracting one from any track over 17; and multiplying the adjusted track number by two and adding the sector number.

Now modify the FAT so that each FAT entry points to the next granule. In the example above, track 20 sector 1 is granule 38, or the thirty-eighth byte of the FAT, counting from zero. Change the thirty-eighth byte to hold 43, the granule encoding for track 22, sector 10. Now change granule 44, the forty-fourth byte of the FAT, to hold 52, the granule encoding for track 27, sector 1. Now go to FAT entry 52, the last granule, and make the last entry of $C5. The $C portion of the code says that this is the last granule of the file, while the numeral five says that there are five sectors in the file. For one-granule files, only this last step has to be performed, to denote how many sectors are in the last portion of the file.

You should now be able to read in the file normally under Basic. Good luck!

There are other tricks that can be done with DDE or similar disk editors, including formatting data as ASCII files, allocating large file areas, and repairing unloadable programs. Get some practice before attempting these operations and remember that DDE is a two-edged utility.

And now for Cesar Frank's famous Second Symphony, the so-called "Computational Symphony." Let's see...oops, I forgot the first violinist in Line 3430...

∎∎∎

Figure 1. Kill Action

```
DRV#: 0    TRK#: 17    SECTOR#: 3
FILE: REC#: 0
OFFSET: C0

00  41  47  49  43  53  51  20  □AGICSQ          THIS BYTE IS ZEROED
42  51  53  00  00  24  00  52  BAS..$.R         FOR A KILL
00  00  00  00  00  00  00  00  ........
00  00  00  00  00  00  00  00  ........          THIS BYTE STILL
4C  41  42  45  4C  53  20  20  LABELS            DEFINES THE FIRST
42  41  53  00  00  25  00  AB  BAS..%..          FAT ENTRY AND
00  00  00  00  00  00  00  00  ........          GRANULE (TRACK 12
00  00  00  00  00  00  00  00  ........          SECTOR 1)
```

ON-LINE

by Wayne Day (76703, 376)

ONE MOST POPULAR FEATURE of the Color Computer Special Interest Group (CCSIG) on CompuServe is the SIG/Access database where you'll find hundreds of programs already programmed, ready to run as soon as you retrieve them from the database. This month, we'll get acquainted with the Access procedures.

Think of SIG/Access as a gigantic electronic library containing books (text files and program documentation), paintings (graphics screens and picture generating programs), records (music files), as well as general programs. Your membership in the SIG is your library card, since no additional fees are charged by CompuServe for access to any of the CCSIG's features.

SIG/Access can be reached by using the XA command either at the SIG menu, or at the FUNCTION: prompt. The system will then ask you which database you want to go to. Select from one of these, depending on your interests:

XA 0 General programs and files
XA 1 Reviews
XA 2 Graphics and pictures
XA 3 Music files
XA 4 Club news and information
XA 5 Reserved for future use
XA 6 OS/9, Basic 09, and Flex

For this example, let's say we're interested in the General program section, XA0, because it contains the largest number of programs and files.

Enter XA at the SIG menu to go into the SIG/Access section of the SIG. You'll then be asked to select one of the databases. Select 0 to enter the XA0 database.

Since CIS accepts commands that are *chained* together, you could have entered XA0 at the main SIG menu to accomplish the same thing, bypassing the second selection menu.

Once in XA0, we need to know what's in the library before we can check anything out, so let's do some exploring.

The CAT command will print out a listing of file names of programs in the database, along with the user ID of the person who submitted them, as well as the date the file was submitted and the length of the file. Files are named in two parts: the file name itself, and the extension, i.e., PROGRAM.BAS. Those users with disk systems are already familiar with the concept of an extension, but for cassette users who haven't had any experience with extensions, just think of them as an additional three-letter description not of the program *itself*, but describing how the program is stored or to be used.

For example, we have a set of standardized extensions for use on the CCSIG:

.CC denotes a Radio Shack Basic program designed to be run on a Color Computer.

Back To Business

Back in XA0 we're still not getting a clear picture of what's available in the database since the file names by themselves can be a bit cryptic, so let's check out the index in a different way.

Enter the command BRO (for BROwse), and you will start receiving a list of all of the files available in the particular database you're in, a complete description of the file, and an option to read or download the file at the end of the description.

If you want to read the file, enter R and the file will be typed out for you (without the system opening or closing your buffer, should your particular terminal program have a settable buffer).

The D (for Download — which means to transfer to your terminal) option will start a series of events that varies depending on which terminal program you are using. For example, users of CompuServe's Vidtex program will be prompted to supply a file name for the program that's about to be saved on the user's tape or disk, and then the program is automatically saved for the user.

If you are using one of the smart terminal programs available for the Color Computer that does not support the CompuServe B protocol (used to guaran-

tee error-free uploading and downloading of files), CIS will send you two special control characters, used to mark the beginning and the end of the file you're downloading.

On COLORCOM/E, for example, the CONTRL-R ($12) character sent by CIS marks the start of an area in the received data buffer, which is then followed by the file itself. This character is used by the program to automatically find the program when you attempt to save it later.

After the file is sent, CIS will send a CONTRL-T ($14) that marks the end of the file. Again, as an example, COLORCOM/E uses this character to denote the end of the file that is being saved.

Most of the so-called smart terminal programs use some sort of control characters to mark the start and end of a file, and the CONTRL-R / CONTRL-T set of control characters used by CIS is the most common, found not only on CIS but on most Color Computer bulletin board systems as well. You might keep that in mind if you're in the market for a terminal program that has a few more "smarts" than Videotex, which most folks start out with because it is sold at all Radio Shack stores.

Since Tandy did not write Videotex to easily support downloading of data to a cassette, there are some special considerations to keep in mind when using Videotex:

● Videotex will not download binary files (those with the extension of .BIN or .IMG)

● 32K Color Computer users should be using Videotex Ver. 1.2, otherwise they will not be able to download files from Access. This is because Ver. 1.0 and Ver. 1.1 of the Videotex ROMpak recognize only a 4K or a 16K Color Computer. If you have a 32K or 64K Color Computer and have Ver. 1.0 or Ver. 1.1 of the Videotex ROMpak, check with your local Radio Shack store about acquiring the current Ver. 1.2.

● 16K Color Computer Videotex users will only be able to download files of up to 6,400 bytes. 32K or 64K Color Computer users with Videotex Ver. 1.2 will be able to download files of up to 22.7K bytes.

Videotex users who want to download files may browse through the files just as do other users. When you decide to download a file there will be a pause while your computer is checked by the CIS system, then you will be asked if your computer has at least 16K of memory. You must have at least 16K of RAM to

download, since a 4K machine doesn't have enough room for the Videotex program and a downloaded file at the same time.

The CIS computers then send the file to you in a form that your Color Computer understands. After the transmission is complete, you'll drop out of the Videotex program and into Basic, automatically.

It is very important that you follow the following steps precisely:

When you get the Basic OK prompt:

● enter EXEC &H8000

● immediately save the file to tape: CSAVE "FILENAME",A

This saves the file in the ASCII format, and is required to have the program saved in a form you can read back into your computer! Do not forget ,A!!

● Turn your computer off for 15 seconds, then turn it back on.

At this time, you'll come back up in Videotex, if you do not have the autostart defeated on your Videotex ROMpak. Hit the Break key a couple of times and you'll be back on-line with CIS at the SIG/Access prompt.

If you have the autostart defeated (see Steve Wegert's article in the March 1984 issue of **The Color Computer Magazine**, page 62) you will come back up in Basic once you turn the power back on. Enter EXEC &HC000 to re-start Videotex.

Should you run into problems downloading with Videotex, check your downloading steps very carefully. Most errors seem to be when folks forget to save the program originally in the ASCII format (i.e., CSAVE "FILENAME",A).

Need more help learning to download with Videotex? There is a special help file on the CCSIG with an expanded set of instructions, located in the X0 datafile. Just enter X0 at the main SIG menu and you'll be able to read the file.

Once you've downloaded a file, regardless of whatever terminal program you're using, you'll see the file name described again, along with the (R D T) prompt. Again, the R lets you read the file, D lets you download, and the T returns you to the top, the SIG/Access prompt.

That should get you started exploring the SIG/Access database system. In the next edition of this column, we'll look at some ways to make your excursions into the SIG library a bit more flexible, and save you some time and money.

I had planned to give a plug for a new Special Interest Group that should be of interest to Color Computer owners: the OS/9 Sig. Tragically, though, the system operator of the OS/9 Sig, Jim Bellomo, died unexpectedly late in February.

Jim conceived the idea of an OS/9 Sig on CompuServe intending a large part of the framework to assist the Color Computer OS/9 user, to assist the user to realize the full potential of this very powerful operating system. The groundwork laid by Jim will be carried on by another Sysop on the Sig, who should be on-line by the time this is published.

Jim Bellomo, age 34, will be missed not only by those who knew him personally, but also by those who met him only through their computers and terminals.

The OS/9 Sig will continue to offer a specialized meeting place for those interested in meeting other OS/9 users, as well as hosting the membership library of the OS/9 User Group, Inc., which will be available only to those members of the OS/9 User Group. An extensive public domain library is in place. To access the OS/9 Sig, go to page PCS-18 on CIS, or enter the command R OS9UG from the CCSIG menu or FUNCTION: prompt.

SIG Suggestions

Finally, a reminder about your CIS User ID and password.

The User ID and password you have is your key to the system, and just as you wouldn't leave the key to your house where anyone could get at it easily, you should also protect your password.

Never give your password except when you first log into the system. If for some reason you get "bumped" from the system, the safest thing to do would be to hang up and re-dial your access number.

Additionally, we recommend that you change your password frequently...once a month is not too often. Simply go to page CIS-175 and follow the prompts.

Remember, CIS holds you responsible for the security of your password, so guard it carefully.

Should you have problems accessing the SIG, or questions about CompuServe's operations, you may call the CIS Customer Services office at (800) 848-8990 between the hours of 8 am and midnight (Eastern time), or leave a message for me on the SIG.

Enjoy the summer! ■ ■ ■

Ed. Note — Wayne wrote this column just after the birth of his first baby — a boy. Congratulations, Wayne!

O-Pak

Frank Hogg Labs
Regency Tower
Suite 215
770 James Street
Syracuse, NY 13203
(315)474-7856
OS-9, 64K
$34.95

O-PAK OFFERS THREE improvements to the OS-9 operating system. It lets your television set display many more columns and rows of text and block graphic symbols than the standard screen format. It has a character set editor for the dynamic creation or change of fonts, and it has a set of conversion utilities that accomplishes the bi-directional transfer of text files between OS-9 and Flex and between OS-9 and the Radio Shack disk operating system (DOS).

Microware Systems Corporation designed OS-9 and defined its standard output device on the Color Computer to be a screen of 32 columns and 16 rows. This specification allowed fast scrolling and the conservation of memory space because it was implemented using the alphanumeric mode and internal memory of the video display generator chip. Unfortunately, it also resulted in compromises which reduced the utility of OS-9. The standard screen limits many serious applications because of its smallness and its inverse (light on a dark background) video rather than true lowercase characters. These limitations are unpleasant for word processing applications since what appears on the screen looks different than what gets printed. Rather than changing the Color Computer hardware, Frank Hogg has introduced in O-Pak an improved screen driver that actually plots all text and graphic symbols on a grid of 256 by 192 pixels (dots). Now you can display information in formats like 51 columns by 24 rows, 64 by 27, and even 85 by 24. One simple command call from the OS-9 Shell is all that's required to change to a high resolution screen for text and block graphics:

OS9: HiRes [<character set name>]

There are twenty character sets, all variable in terms of character width, height, and foreground and background color.

Color options are the same as those OS-9 allows in its graphics mode. Each character is displayed within a cell. The cell is usually larger than the character to allow for borders between adjacent characters. With a special O-Pak command this space can be reduced or eliminated. O-Pak permits spontaneous changes to the amount of vertical and horizontal cell space, too. Since inter-character spacing and inter-line spacing can be set independently, more characters and more lines can be placed on the screen virtually at will. Some sets, the very dense ones, are not readable on the screen, but have been supplied because they are useful for showing page format — which areas of a page contain information and which are blank.

There are also character sets that are rich in block graphic symbols, sets that are convenient, versatile tools for producing simple art. Graphics can be mixed with text since HiRes is in the graphics

> "... a new concept is appearing... a window acts like a smaller screen. O-Pak has up to eight windows..."

display mode. Consequently, it's possible to construct, say, a fancy form on the screen that contains a logo, italicized field names, horizontal and vertical lines to separate rows and columns, and a simulated signature. Imagination is the only limit to this capability. Furthermore, if you had an Okidata Microline 82A printer and used the graphic symbols from the character set "mi82a," a close replica of the form on the screen could be printed on the Microline 82A. Suppose you have a different printer with different block graphic symbols. Described later is a technique that helps you define these block graphic symbols (in fact, any of the characters) for display on the screen.

Three other features of the HiRes screen are scroll rate, cursor selection, and windowing. Since the screen is fully controlled by software, display scroll speed can be changed at will from as little as one pixel at a time (smooth scrolling) to as much as 16 pixels at a time (jump scrolling). The style and blink rate of the cursor can also be selected. The cursor may be either an underline symbol or a block symbol, and may be continuously displayed or blinked. The blink rate may vary from $1/60$ of a second to over 40 seconds.

Windowing

Windowing, a relatively new concept in the computer field, is making its appearance on rather expensive microcomputer systems. A window is part of the screen that acts like a separate, smaller screen. There can be up to eight windows on an O-Pak screen, and each may be scrolled and manipulated independently. Each window has the line width defined by the character set and is divided from an adjacent window by an imaginary horizontal line.

One practical application for two windows is in word processing. Suppose you want to keep a help menu at the top of the screen while text is being developed below. The top area would be a small window that would be changed infrequently. The larger window below would contain text to be scrolled and moved as needed. There would be no need to re-create unchanged information when another area of the screen had to be changed. I expect to see many clever applications of the window concept for the Color Computer as HiRes gains wider acceptance.

Function Control

The many different functions of HiRes are controlled by the forwarding of special code bytes to the standard output path. For example, the hexadecimal character sequence 1B 43 creates a new window. An application program merely directs the appropriate code bytes to the standard output path using an OS-9 system call for line output. At the command level, it is customary to send code bytes from the shell by way of the Display command, such as:

OS9: DISPLAY 1B 43

Here is a summary of control functions for HiRes. OS-9 itself also defines controls for alphanumeric and graphic modes. Those controls are complementary to the HiRes functions and may be found in the *OS-9 Commands* manual from Tandy. Display Control Functions are:

1B 41: erase from cursor to end-of-line
1B 42: erase from cursor to end-of-window
1B 43: create a window boundary above current line

1B 44: combine current window with one above

1B 45: insert a line at cursor

1B 46: delete the line at the cursor, compress screen

1B 47: shift current line right by 8 character widths

1B 48: shift current line left by 8 character widths

1B 49: shift cursor one pixel right

1B 4A: change to character set whose name follows

1B 4B: shift cursor one pixel up

1B 4C: shift cusor one pixel down

1B 4D: shift cursor one pixel left

1B 4E: revert to a single window

1B 51: enable overstrike mode

1B 52: change to specified foreground color

1B 53: change to specified background color

1B 54: change spacing between characters

1B 55: change spacing between lines

1B 56: select block cursor

1B 57: change to specified blink rate

1B 58: change to specified scroll rate

1B 71: disable overstrike mode

1B 76: select underline cursor

1C: ignore all control codes but 1E 1E 1E and 1F 1F 1F

1E 1E 1E: accept all of the above control codes

1F 1F 1F: terminate HiRes and resume with standard 32x16 mode

Character Sets:
What You See
Is What You Get

If you do not find a character set to suit you, you can make your own with CSEdit, a utility that changes any of the given character sets and can add new ones. The beauty of this editor is that it is not necessary to sketch each character on paper, decompose it into pixels, and translate the pixel pattern into hexadecimal numbers. That old, cumbersome method now seems as archaic as finding square roots with pencil and paper instead of a calculator.

CSEdit draws a large image of the cell that contains the symbol to be modified. The cell is shown as a mosaic of squares where each pixel of a symbol is represented by one shaded square. Unshaded areas represent blank areas, like the space inside a character or between adjacent characters. The cell is drawn on a grid with as many horizontal positions as the width of the cell and as many vertical positions as its height. Every pixel can be individually selected by positioning a cur-

Color Computer/127

sor at its location. Shading is toggled on or off by keyboard control. Presto, instant customization.

Character editing is fun and gives me kind of an artistic feeling, too. Next to the mosaic of the character is the same character shown in normal size. As squares are toggled on or off in the mosaic, the mirror image in normal size changes correspondingly. I had no doubt about how the final edited version would look because it was right there before me. From now on, what I see on the screen is what I get on the printer.

CSEdit uses several menus to simplify setting the editing options. A six-choice main menu establishes a "home base" from which more detailed menus are called. The choices let you edit a character set, write a file, write a file and exit, abandon current work, execute a shell command, and change the HiRes mode.

During editing of a character set the edit submenu offers the choices of changing cell and character width and height; changing color set, foreground color, background color, and graphics mode; changing character shapes (by the technique described above); renaming the module which contains the character set; and returning to the main menu.

When writing a file users are prompted for the file name to save the character set as a data module. The revision and edition numbers of the module may be changed. (The revision number is important if this new module is to dynamically replace an older module in memory). Then the file is written.

In the write a file and exit mode, after getting a file name and writing the file, CSEdit returns to OS-9.

The abandon current work option is a graceful way of retreating from your own disaster.

Any shell command may be called from within the edit process. After executing a shell command, editing control is restored as though no interruption took place.

When changing the HiRes mode users are prompted for changes to make to the character set in use, not the set being edited. Subject to change are the character set itself, foreground color, background color, graphics mode, colors, height, and width.

File Transfer Commands

Before O-Pak file utilities, there was no convenient method for inspecting a file on a Flex or Radio Shack disk. There was no way to convert a text file or to transfer a file to another operating system. These limitations have been removed with the arrival of four new OS-9 commands: XDIR, XDUMP, XLIST, and XCOPY. They operate on files created by the Radio Shack DOS, Flex, and OS-9.

Each command satisfies a fundamental, unadorned service requirement. XDIR displays on the standard OS-9 output device the directory of a designated Flex or Radio Shack disk. XDUMP produces a combined hexadecimal and ASCII file dump on the standard output device. Similarly, XLIST lists a text file. XCOPY is more versatile. It can convert and transfer a text file from a Radio Shack disk to an OS-9 disk, from a Flex disk to an OS-9 disk, or from an OS-9 disk to a Radio Shack or Flex disk.

The O-Pak file utilities need parameters that specify operating system file types. A new separator (the percent sign) and three new prefixes are used to designate target or recipient file types: Type rs% for any file in the Radio Shack DOS format; flex% for text files with space compression in the Flex operating system; and flexb% for uncompressed (binary) files in Flex. No special prefix for OS-9 is necessary.

Documentation

The O-Pak manual has 42 pages of well written, well organized information. It even stacks neatly with OS-9 manuals. It has four chapters, two appendices, and an index. After an overview of features are installation instructions and documentation on HiRes. Implementation strategy is summarized and, since a nonstandard approach was elected by the software designer, an understanding of that section is essential if you are to create new master OS-9 disks. The next chapter describes the character set editor. It can stand alone but is most useful as a companion to an interactive session on the Color Computer. Chapter four describes the file utilities, and appendices list display control codes and a glossary of terms found in the manual.

O-Pak text files also contain good reference material and probably should be listed on your printer for future study should you buy this product. There is also a description of each of the twenty HiRes character sets on the disk.

My review copy of O-Pak, Release 001, dated November 30, 1983, is quite robust and behaves correctly in nearly all instances. The only errors I found were in the processing of responses to two menus. Here, my clumsy fingers and malicious responses caused OS-9 to lock up. If you have this release of O-Pak, be careful not

to hit the Break key while entering data in the menu to change character width. Attempting to change the graphics modes from the HiRes menu also creates a lock-up. You might want to use the alternate method with the Display command instead of the editing function. It would have been better if O-Pak had rejected invalid responses.

Performance

When you use O-Pak you trade free memory space and speed for high resolution displays and versatility. The speed at which the display scrolls is affected in the HiRes mode since 6144 bytes of symbol data must be manipulated by software rather than automatically by the computer hardware. Although HiRes allows various scroll rates, in my opinion the speeds range from "slow" to "very slow." For most applications, I find this "slow" pace is rather comfortable, though. If speed is a temporary concern, HiRes can be turned off and recalled later.

File commands also suffer from largeness. They are considerably larger than HiRes because they must recognize and process disk and file formats for three operating systems. These are not small tasks. Respectively, the module sizes of XDIR, XLIST, XDUMP, and XCOPY are 14659, 14862, 15133, and 20151. These commands take a large percentage of memory space and use many disk sectors. I would not indiscriminately load these commands in memory or place them on my master work disk. Forethought and planning is in order if you are to make the most of this system.

High resolution displays may not look very clear on a standard color television set or even on a high quality color set that's being used as a monitor. Expect to see multi-colored ghost images of the characters. The difficulty stems from video circuitry, not from O-Pak. To improve readability, turn the color control on the TV set down. Clarity is improved even though you lose your background and foreground colors. If you are using a monochrome display, this problem does not exist.

If you are the type who plans to make a copy of OS-9 or to modify the kernel (guts) of this operating system, beware. HiRes should be turned off before performing a Cobbler command. CCIO must be in its restored state before requesting a Save of its memory module. Failure to follow this precaution will result in a CRC error in the CCIO module and will prevent a correct boot of the new kernel. If future releases of OS-9 change the CCIO mod-

ule an updated release of O-Pak will, I suspect, be necessary to keep in step.

O-Pak is among the most useful additions to my OS-9 library. This package has transformed an unattractive 32x16 screen to a display medium of convenience, even beauty. Perhaps my highest praise for HiRes is that I take its features for granted now and rarely need to be aware of its presence. Add to that the value of transferring files to other operating systems and you can see what a powerful collection of tools I now have.

— *Eugene Grunby*

Autoterm

PXE Computing
11 Vicksburg Lane
Richardson, TX 75080
(214)699-7273
**$39.95 cassette, $49.95 disk
32K**

*I*N TERMS OF INTELLIGENCE, Autoterm is a cut above many other smart terminal programs. It does everything its ads say it does, too. What sets it apart from greatness are a couple of documentation problems and operational kinks. Still, it's a fine program that can do even more than what it was designed to do, if you've got enough imagination.

The program requires at least 16K of RAM, which yields a storage buffer space of slightly over 2K. A 32K Color Computer, however, lets the program store up to 18,474 bytes in the buffer, a much more usable amount.

The review copy program came on an auto-run cassette, so it overwrites the reset vectors within the computer's RAM, and you do not have to enter EXEC or GO ... it starts up by itself. The disadvantage to this approach is that you can't escape the program without turning off the power, nor can you make a back-up copy very easily.

Autoterm supports a wide range of baud, parity, full or half-duplex, and stop bits, making it adaptable to a whole spectrum of host computers. In the terminal mode, you may choose between several formatting options, and have the ability to turn on the "key-beep" that "plinks" each time you press a key, or the "key-bop" that lets you know when you've made a mistake.

Word wrap, the ability to prevent a word from being broken in half at the end of a line, may be manually enabled or disabled, as can a reversal of the normal screen of black letters on a light colored screen to light green letters on a dark screen.

Printers are supported off line, though with an RS-232 switcher the modem can be left connected to the phone line while Autoterm routes something to the printer, and you don't have to restart Autoterm if the carrier drops for some reason. Several options make using this program a joy. One is the special printer driver, which embeds printer control codes to modify how the printer acts. For example, you can change from normal print to double width print.

One unique feature is the program's ability to override the length of a line of received text and re-format it to whatever application you want. Since the format of a line from a bulletin board system (BBS) isn't often the length your printer can print on one line, you can save some paper with this feature.

A simple text editor is included in Autoterm. It may be used to prepare messages off-line for later transmission, or to generate or modify a series of user-programmed text files known as Key Stroke Multipliers (KSMs). With a KSM you can set up a certain procedure, such as logging on to CompuServe or another electronic message service, and have the Color Computer do most of the work for you, like entering your name and log-on information, automatically reading messages, and saving them.

A large number of sample KSMs are in the instructions, including one which lets you program Autoterm as a mini-BBS. That one lets you get electronic mail even when you aren't home. The possibilities of such applications are endless.

Overall, Autoterm is a valuable addition to the Color Computer's communications ability. It has its own problems though.

The instructions fail to mention that when the buffer becomes full, the screen display changes by dropping all its blank lines. And the documentation does not explain that the buffer doesn't overwrite itself once it becomes full, erasing the oldest material in the buffer like most terminal programs do. Instead, it requires you to delete a portion of the buffer to make room for new data.

Take away another point for the lack of any means for the host computer (the one you're calling) to automatically open and close, or even mark, a portion of your buffer when downloading. Most

Color Computer bulletin boards have adopted the defacto standard of the CONTROL-R / CONTROL-T protocol (hex numbers 12 and 14) to mark the beginning and end of a downloaded file. Instead, you must manually go through the buffer and mark the start and stopping points yourself.

Finally, a warning to CompuServe users: don't be surprised if you get an Invalid Password error message if you're a fast typist when you log on to the system. Autoterm waits one-fifteenth of a second for an echo from the host computer before it sends out the next character. Fast typists may be typing passwords faster than they're being sent out, since Autoterm is waiting to get something back and the password is not echoed back to your computer. The cure is to slow down, at least when typing your password. This lesson was learned after quite a few hours on the phone with CompuServe trying to figure out why my password wouldn't work any more.

This program is protected but all is not lost since PXE offers something that may make other software distributors sit up and take notice. For an additional $23, PXE will sell you a "registered" and unprotected version of Autoterm, a work-alike version of the regular program except for the following differences.

The auto-run leader is not present, so the cassette version may be backed up, or loaded onto a disk.

The program is written in position independent code, meaning you can relocate it anywhere within your available RAM. Using PXE's instructions, a 64K Color Computer can wind up with about 39K of buffer.

"The possibilities ... are endless."

A set of technical notes accompanies the program, including a list of all jumps and calls to external routines, making modifications and customization much easier.

To get the "registered" version of Autoterm you must send in the application enclosed in the original Autoterm documentation. That way only authorized legitimate purchasers of Autoterm may buy the unprotected version.

— Wayne Day

TRS-80 Color Mouse

Radio Shack
One Tandy Center
Fort Worth, TX 76102
(817)390-3885
$49.95

THE COLOR MOUSE is a product intended to make game playing and graphics creation faster and easier. The Mouse connects to either joystick port and can be used with any program that supports a joystick. It also can be used with any Color Computer no matter

---

which version of Basic or how much RAM size you have.

To use the Mouse, you push it around on any tabletop, within a 4.5-inch square. At the top left of the square the computer will read an X of 0 and a Y of 0. At bottom right the computer will read an X of 64 and a Y of 64. Other points in the square have appropriate X and Y coordinates. In that respect, the system is essentially the same as the joystick system.

To envision how Mouse works, imagine a trackball like the one on arcade versions of Centipede or Missile Command. Encase the trackball in a small black box, flip it over, and put a fire button on top. The trackball is spun by moving the box. The Mouse follows this basic plan, but its trackball is smaller and the cable comes out the back.

The Mouse is 3.5 inches long, 2.5 inches wide, and 1.3 inches high. The cable is five feet long with a joystick plug at the end. On top is the bright red bar of the fire button. The bar travel is about 1/16 of an inch and clicks when it makes contact. The case is the same textured black plastic used on regular joysticks.

A look inside reveals sturdy construction, including a well-anchored cable, good solder connections, a steel track-

ball, and a heavy-duty momentary contact switch for the fire button.

The manual is a ten-page booklet that tells you how to set the Mouse up, how to use and care for it, and what its technical specifications are. It also suggests three Radio Shack program pak games the Mouse can make easier; Wildcatting, Polaris, and Galactic Attack. I don't have Wildcatting, but I did try this product on the other two. I found no improvement in Galactic Attack. Polaris, however, became a totally different game. I controlled the cursor much more easily, not only because of the larger movement area, but also because I could use one hand on the Mouse and the other on the keyboard.

The Color Mouse has three disadvantages. First, the cable tends to get in the way. I place my disk drive to the right of my computer and the Mouse cable tends to get caught under it. Second, it is nearly impossible to tell what the Mouse's X and Y coordinates are without a screen reference. To help, I drew a 4.5 inch square on a piece of paper and used the Mouse on that. Last, split-second movements sacrifice accuracy. The steel ball will slip instead of roll on the table when it is jerked or when directions are changed

quickly, so the Mouse simply can't match joysticks for speed. And the 4.5 inches of placement freedom make large fast movements necessary with some games, much more so than with the 2 inches of freedom allowed with joysticks.

There *are* things that *can be done* more easily with a Mouse than a joystick and vice versa. The Mouse is capable of very accurate and predictable placement, while joysticks are for registering reflex motions. But the Mouse does pretty much what Radio Shack says it will do. It's well-built, rolls smoothly, and adds zest to some games.

— *L.R. Jansen*

IntBasic

Spectrum Projects
93-15 86th Drive
Woodhaven, NY 11421
(212)441-2807
16K to 64K
cassette only
$39.95

*I*NTBASIC IS AN INTEGER Basic compiler program written to run under Color Basic in 16K, 32K or 64K machines.

It compiles programs into machine code so they can run faster than with the Basic interpreter. It uses many of the familiar Basic commands and the usual mathematical and logical operators. A new command, IBSHFT, lets you shift the bits in any variable to the left or right. This command is functionally similar to the 6809 assembly language LSL and LSR operators.

As its name implies, IntBasic works only with integers. If you need to work with floating point numbers you will have to write routines that keep up with exponents and signs. IntBasic allows 26 integer variables, A to Z, each of which can be a 16-bit value from 0 to 65,535. It also allows 26 dimensioned "string" variables A$( ) to Z$( ). Each string variable is an array of one-byte integers with values from 0 to 255. For example, A$(5) refers to the fifth byte of the A$( ) array. The length of each array is set by the Dimension (DIM) statement. IntBasic does not distinguish data types, so integer and string variables can be manipulated and compared in the same statement; the value of the specified byte of the string variable and the lower (less significant) byte of the integer variable are used. For example, the assignment statements A = B$(35) and F$(10) = C are valid. The usual string operators RIGHT$, LEFT$, MID$ and INSTR cannot be used. If you need these functions, you will need to write your own routines.

Now, this sounds like work. But if you want the speed of machine language you have to sacrifice some of the Basic's conveniences. You will need to learn "memory management." Specifically, you will need to plan where in memory to put the compiled program and a table for any text strings. You must be careful not to overwrite any portion of the original Basic or compiler programs. In addition, a library of subroutines in the compiler for multiplication, division, input and so forth must be in memory when your compiled program is executed. More planning is involved than usual, but the experience will be good if you want to learn assembly language programming.

IntBasic comes on cassette in 16K, 32K and 64K versions. The memory requirements for the compiler vary from approximately 13K to 22K. Unfortunately, the 16K version leaves only 1764 bytes for the original Basic program, the compiled program and the text table. The major disadvantage of IntBasic is that it is not a disk based system. I had almost forgotten how much I hate to find and load programs on tape. IntBasic also does not perform disk or cassette input or output functions. The users manual says you can tailor the existing ROM routines to perform these I/O functions.

The manual, a 19 page document, describes clearly and thoroughly everything you need to know to use the compiler. It is not an easy read because of the moderate complexity of the subject. A table of contents would help. It does contain four sample programs that can be used to learn the compiler. The first two can be run with 16K (although they are examples for a 32K machine) if the compiled program starts at 2000 (decimal) and the text table starts at 2500. Line 24 of the second program should be corrected to read: FOR J = 2 TO C: I+I+1: Z=PEEK(I): X=A$(J).

During compilation, the compiler can generate listings on the screen and line printer to reproduce the original Basic program, and give the memory location of: each line of the compiled program; the first and last address of the text table, compiled program and variable table; and the start address of every integer and string variable.

— Raymond D. Mosteller

Circle No. 76 on Reader Service Card

Please indicate which of the following microcomputers you currently own and/or plan to buy in the next 12 months.

	1 Own	**2** Plan to Buy
Apple	A	L
Atari	B	M
Commodore/PET	C	N
Digital Equipment/DEC	D	O
Heath/Zenith	E	P
IBM	F	Q
Radio Shack/Tandy TRS-80	G	R
Color Computer	H	S
Other Tandy/Radio Shack	I	T
Other (specify)	J	U
None	K	V

3 For what, if any business application(s) do you use the microcomputer you currently own?_____

Circle Number   Void after August 31, 1984

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108
109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162
163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198
199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234
235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252
253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270
271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306
307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324
325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342
343 344 345 346 347 348 349 350

CLC6843

PLEASE PRINT CLEARLY—Use only one card per person.

NAME_____ PHONE NO. (____)_____

COMPANY_____TITLE_____

ADDRESS_____APT._____

CITY_____STATE_____ZIP_____

4 ☐ **Please send me 12 issues of The COLOR COMPUTER for $17.97 and bill me. (Full 1 year subscription price $24.97.)**

4 How do you use your COLOR COMPUTER?
w. education
x. entertainment
y. personal purposes
z. business purposes

5 Which of the following peripherals do you plan to buy in the next 12 months?
1. Monitor
2. Disk drive
3. Printer
4. Modem
5. RAM chips
6. Keyboard
7. Interface Board

6 Where do you buy software?
8. Mail order
9. Radio Shack computer center
10. Radio Shack store (not computer center)
11. Other computer stores

7 How many software packages do you plan to buy in the next 12 months?
12. 1-2
13. 3-5
14. 6 or more

BUSINESS REPLY MAIL

FIRST CLASS   PERMIT NO. 27346   PHILADELPHIA, PA

POSTAGE WILL BE PAID BY ADDRESSEE

THE Color Computer....
MAGAZINE

P.O. Box 13307
Philadelphia, PA 19101

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

Timebound

by Children's Computer Workshop, Inc.
Radio Shack
One Tandy Center
Fort Worth, TX 76102
(817)390-2842
16K, cassette
joystick
$19.95

*T*HE CHILDREN'S TELEVISION Workshop has filled our television screens with quality educational programming, fun, excitement, and challenge, for 13 years. So the creation of interesting educational software from its subsidiary, the Children's Computer Workshop (CCW), is hardly surprising. Timebound is part of that branch of the CCW family Tandy calls home-based educational programming; it's aimed at parents and children more than teachers and students.

The goal of any CCW game is to "encourage children to experiment and explore, to think and solve problems and to practice skills while they are having fun." Timebound, part of CCW's "Creative Exploration Series" for ages 10 and up, fulfills that goal but not in the cutesy tradition of Sesame Street nor with the comic book antics of The Electric Company. It's more reminiscent of the 3-2-1 Contact's intrigue.

Timebound offers creative exploraton of a time loop between the years 0 and 2000. The game booklet is well-written and to the point, and within minutes I grasped my role as a scientist who must rescue a timebound assistant, Anacron. Despite my warnings to stay away from the Time Machine, Anacron didn't. He went tumbling out of control, frantically grasping historic events in an attempt to return to the present. I have only 15 minutes to retrieve him or my assistant will be timebound forever! So:

I am stationed at the Time Machine's viewing screen and in control of the Time Scanner (joystick). I check my starting year for this particular game on the calendar at the top of the screen — 1230. I take a deep breath, brace myself for action and begin traveling through time at high speed. The calendar changes rapidly as I pass through time bands. Events flash by, blurry with speed, as I move my scanner around the outer band hoping to catch the event Anacron is clinging to.

My scanner catches an event and the view screen decides to freeze. The name of the event appears at the top of the screen and the calendar tells me what year I'm in. Anacron's location appears at the bottom of the screen and my heart starts beating faster! My event is "bicycle," the year is 1816, and Anacron is at "compass." The view screen divides the events in history into 11 categories so I check in the Timebound manual. I decide I'm in the "land transportation" category and that Anacron is in "scientific tools." I make a mental note of the position of the time scanner so I'll know where "land transportation" is. Category positions change with each game and I must discover and remember them so I can find Anacron's location. Now I concentrate on time sequence. Did we have compasses before we had bicycles? I feel pretty sure we did so I press the red button on the joystick to resume the search. I also press the spacebar to change direction.

Suddenly I'm plunging backward in time. Years are melting away as I frantically rotate the scanner around the view screen. The clock continues to display elapsed time. I've already used nearly five

minutes! I catch one event after another as I memorize the category positions and link events with their dates in history. I approach the year 1125 and hear a "beep." My scanner catches an event and the view screen freezes action once again. My heart sinks. I've caught Anacron's event, but as I've moved closer to him I've created a time wave that has caused him to lose his grip. I have "compass" in the year 1125 but Anacron is no longer there. He has tumbled to another event, perhaps even another category. I've used over nine minutes now.

I press the red button and the years fade back to 0. I hear a series of sound effects and the year 2000 appears. I'm through the time loop now. As the 1990s pass I catch another event. It's 1990 and I'm on "hot tub tennis"... no kidding! Now that's something to look forward to. Huh? Oh yes! Anacron, hang on, I'm on my way!

Timebound can be just that exciting in the hands of the right person, young or old. But since the graphics are limited and unvarying, and the reinforcements simply faster personal times and achievement of each of the 11 difficulty levels, I recommend Timebound for *very* curious children with above average attention spans. And since memory and reflex action will be developed while playing Timebound, beginning play with at least some skill in those areas is important.

Catching an event with the scanner is a matter of luck of being in the right place at the right time. Granted, the point of the exercise is to develop a sense of history, but I'd still like some control there and I'd also like some occasional hints when I get within a reasonable range of Anacron. If you don't know what an event is, you have no way of knowing how close to Anacron you are and you're left guessing through centuries until you catch a familiar event. And, when Anacron is on an event you're not familiar with you must simply play blindly until he changes events or you accidentally stumble upon him.

The higher the game level, the more categories Anacron will use and the faster the scanner will move. When you catch Anacron he appears on the screen as a large question mark with eyes, at the appropriate category position and with sound effects. If you fail to catch him in 15 minutes you'll hear a sound effect and the screen action will freeze. In either case, a new game can be started by pressing the Clear button.

The categories that divide the game's historical events are: air and space, time, communication, land transportation,

water transportation, architecture, fun and games, scientific tools, political leaders, household items, and sports. If you want to learn about a specific category you can keep the scanner in that particular spot on the screen to catch all the events offered in that game. Of course you may not catch Anacron that way, but it's a good way to get familiar with Timebound.

I enjoy playing Timebound but, more importantly, my ten-year-old son enjoys it, too. In fact, he read the manual on his own and was soon finding Anacron faster than I.

Many names and events included in the game are a mystery to young minds, though. Genghis Khan, Hadrian, Eniac, and Carrrack, for example. Even things like a pressure cooker, candle clock, and sextant can be strange to them. My son learned to hold the scanner to catch an unrecognized event until further events identified the category. That solved some of the mystery. A dictionary and encyclopedia satisfy extreme curiosity when a game ends. Occasional adult participation will help children learn unfamiliar categories and will encourage activities like the riddle-solving at the end of Timebound's manual.

How well I remember agonizing for long, boring hours, trying desperately to memorize events, names and dates for school exams. Timebound... welcome.

— *Jean Plesser*

Factory

Sunburst Education
39 Washington Avenue
Pleasantville, NY 10570
32K disk
$39.95, regular
$49.95, teacher's edition

*A*S A COMPUTER-USING educator, I am constantly urging children to attack a large problem by breaking it down into small parts. By solving each part step by step, the larger problem can be solved. The Factory by Sunburst is a superb way to teach this problem-solving strategy. It requires mental dexterity, careful observation, and planning.

There are three levels in Factory. In the first, called "Test a Machine," students experiment with the three machines that are used in the factory. Students' creativity can emerge in the second level, "Build

a Factory," as they design products, program an assembly line, and challenge other students to produce the same "product." In the third level students are asked to work backwards and determine what processes were needed to construct a product. In order to do that, they must know the function of each machine and devise a sequence that can make that product.

Test a Machine, level one, lets the kids try each machine to see what effect it has on the raw material, a plain, blank square. The three machines either PUNCH, ROTATE, or STRIPE. Each machine has its variations, too. For instance, ROTATE will let you turn the square 45, 90, 135, or 180 degrees before sending it on down the assembly line. Colorful graphics depict the machines in operation, and sounds are made each time actual "work" is done.

The second level allows up to eight machines to be set up in an assembly line. If the students PUNCH a square in the center, ROTATE it 45 degrees, and put a thin STRIPE on it, the finished product will be a diamond-shaped disk with a diamond hole in the middle and a horizontal stripe running through the mid-line.
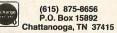
When students finish creating an assembly line, the gears in the factory move and the finished product is graphically produced. Then the computer asks if they want to challenge others to make their product. Students delight in devising elaborate products to challenge their friends.

In the final stage of Factory, students see a product that has been made by several machines. They are asked to reconstruct the sequence of machines and processes its creation required. The computer gives the choices of easy (two machines), medium (up to four), and hard (up to eight). Students must think backwards and assemble the required machines, then the product will be put through the new assembly line. It appears on the screen next to a depiction of the original. If the student-devised product varies from the original the computer declares, "YOUR PRODUCT HAS A FLAW." Students can then try again or start a new product.

There is no limit to the number of times the students may attempt reconstruction. This feature encourages the young engineers to take careful notes, learn from their mistakes, and discuss possibilities with others. At any time they can stop what they're doing and return to the main menu. The BACK UP choice is also constantly available, allowing them to return to their previous choice. Documentation is complete and useful.

One sixth grader suggested that Factory would be a good training device for the Olympics of the Mind Competition, a nationwide creative problem-solving event now in over 3000 schools. Many problems in this competition are of the mechanical type. Students must learn that mechanization is, in actuality, a series of parts with each part assigned one specific task. Factory teaches this concept as well as any activity I've seen. The versatility of Factory satisfies my expectations for good simulation. It can be used by fourth graders and adults, and is applicable for individual and for group instruction.

— J. Craig Dickinson

Video Plus

Computerware
Box 668
Encinitas, CA 92024
(619)436-3512
$24.95

*T*ELEVISIONS ARE FINE for many computing applications, but if you do much word processing you probably suffer from occasional eye fatigue and headaches. A high resolution monitor is much better to work with, especially when you're using programs that generate video displays of up to 85 characters per line.

On a television, displays above 51 characters per line are next to impossible to read. With a high resolution monochrome monitor, such character display, even 86 character displays, can be more easily read. Unfortunately, one of the biggest drawbacks of the Color Computer is the fact that it does not, in its standard configuration, easily lend itself to a marriage with a high resolution monitor.

The problem arises when you try to connect a high resolution monitor to the Color Computer's video jack. That jack was designed for output to a television set and therefore uses radio frequencies. It will not work with a high resolution monitor. In order to use a monitor you need raw signals from inside your computer. Several articles that describe exactly how to do that have appeared, but they are technically oriented and require some degree of expertise to implement. I had almost resigned myself to the idea of living with unhealthy television displays when I heard about Video Plus.

Video Plus is a hardware device that gives you a connection for monochrome or color high resolution composite monitors. I was skeptical of the "installs in minutes" claim, as I have heard that one many time before. Video Plus, to my surprise, lives up to the claim.

I timed my installation, beginning with the first reading of the installation instructions. It took me 11 minutes, including time to read the directions twice, take the cover off the computer, install Video Plus, and reinstall the cover.

The instructions are completely clear and the package contains a preassembled and tested unit. To install, you simply remove one IC chip (easily identified in the instructions), install a socket and replace the chip in the socket. All the wiring and soldering has been done for you. Next, you simply stick the small Video Plus board (it comes with a self-adhesive bottom) somewhere convenient and you are finished. All cables are included and are prewired. Most of your thinking will center around finding a convenient place in the case through which you can route the cables.

Video Plus comes with a sound patch too, so you can capture signals from software that generates music, speech or other sounds.

When you are finished with installation you have about two feet of video cable with a male RCA jack on the end that plugs into your high resolution monitor. You also have another cable with a female RCA jack that connects the audio. Since most monochrome monitors do not have speakers, you may have to capture the audio output with an external amplifier such as a tape recorder or

"I leave my work station with eyeballs clear and wits intact."

radio, using an auxiliary input jack. I use a small, battery operated amplifier. You'll also need a patch cable with a male RCA jack on one end and a standard phono plug on the other.

All right, all right, I can hear you. You're saying, "So far so good, but what about our COLOR!?" Well, if you want to use a high resolution color monitor, you can, even though Video Plus comes set up for monochrome. What it normally does is strip the monochrome signal and

eliminate "irrelevant" color signal information, since monochrome monitors have no need for it. That gives you an outstanding monochrome display, complete with all shades of grey. If you want color, you cut one wire loop to preserve the necessary signals. If you wish, you can wire a small switch to the wire loop and locate it somewhere that lets you switch back and forth between color and monochrome signals.

The unit includes an adjustment that lets you find the optimum signal for your monitor. Another plus is the fact that your standard TV output is not disabled.

Video Plus is an outstanding product that does everything it lays claim to. I now enjoy the sharp, clear, and interference-free display that is possible only with a monitor. And I leave my work station with eyeballs clear and wits intact.
— Norman Garrett

Mailing List

Prickly Pear
9234 East 30th Street
Tucson, AZ 85710
(602)886-1505
32K, disk drive
$49.95, disk only

*I*F YOU HAVE NEVER used a label-making data-base program, Mailing List may be a good place to start. It's friendly and comes with instructions to customize it to your particular computer-printer combination. It is, however, less than flexible in setting up fields, and slow in sorting.

The main reason it's friendly and easy to customize is that it is a combination of two Basic programs and one assembly language program. The Basic programs are Mail and Alpha; the assembly language program is Screens, a high-resolution screen program that's automatically called in by the main program, Mail. You get a choice of two screen colors with Mailing List, a light one, which is a white background with black characters, and a dark one, which is a green background with black characters. The dark seems to produce the least amount of eye strain.

To start Mail, all you have to do is type in RUN "MAIL" and hit the Enter key. You will be asked if you intend to start a new file, the file name, and your screen selection. After you answer those questions you reach the main menu. The main menu has seven choices. They are, Start/

Add, Review/Edit, Print All, Print Particular, Print Phone Numbers, Load New File, and End Program.

The data base for this program is composed of six fields; Name, Address, City/State, Zip Code, Phone Number, and Code. One weakness in this program is its inability to have more than one Name

---

"Mailing List is friendly and easy to customize... but slow."

---

or Address field. Many times when you are preparing mailing labels you need to use more than one address line, and sometimes you need to mark a label to someone's attention. That can not be done neatly with this program.

Another hang-up that detracts from Mailing List catches you after you have started a new file and have entered a number of records. At that point you might want to edit them. The program won't let you do that unless you first end it and rerun it. This only happens the first time you start a new file, but it is annoying.

What Alpha does is let you copy or sort the data files produced by the Mail program. Alpha sorts the Name field either by the first or last names. All sorts are in ascending order. To change the sort field you must take the trouble to change Line 400.

To sort addresses you end the Mail program, turn the computer off and back on, type in RUN "ALPHA", and hit the Enter key. This clears the computer and removes the Screens program. Alpha will not run with Screens in place.

The sort is a modified bubble sort. It is very slow. It took 12 minutes to sort 60 records. Also, all sorting is done on the disk, so my disk ran for the full 12 minutes. Prickly Pear acknowledges the sort is slow and suggests that any large sorts (1000 or more) be done overnight. If your disk drive is like mine and is rated at 200 to 300 hours, it's not smart to run it for hours on end.

This program would make a good starter program for someone learning about data bases, but its price is what you'd expect to pay for a professional business program, not one in which you have to change a line in Basic to change the sort field. Perhaps at half the price, this would be a good buy!!!
— Bob Jack

The TRS-80 User's Encyclopedia

by Gary Phillips
and Guier S. Wright III
The Book Company
11223 South Hindray Avenue
Los Angeles, CA 90045
(213)410-9466

Okay, first things first. There are two of these publications, one for the Models I, III, and IV, and one for the Color Computer and MC-10. This announcement regards the latter. It's 241 large pages long and costs $14.95. It's cross-referenced and includes listings of software houses and product information. It's written for beginners and experienced users.

Dragonfly Fan

Dragonfly Software
404 West Stafford Street
Philadelphia, PA 19144

Dragonfly Software is offering its fan "while supplies last." The $18 fan is a solid state unit. While its press release states it has "no wearing parts," it does have, instead of the traditional propeller, oscillating wings. The fan should "last forever" — some have run for over 30,600 hours, Dragonfly says.

Computer Novelties

Sweet Gum, Inc.
15490 NW Seventh Avenue
Suite 204
North Miami, FL 33169
(800)237-9338
in Florida (305)687-9338

The Sweet Gum Catalog is an offering of novelties for computerists. Messages on the T-shirts range from "Byte My Bits" to "I'm Bilingual, I speak English and Basic." The message on the mug changes when hot liquid is poured in from "computer expert on duty" to "expert off duty." There's a wall clock with binary numerals, and a calculator chocolate bar. A microchip-patterned tie, a chip off the old block Lucite paperweight, a 600-piece puzzle of a microchip... Sweet Gum will send you the catalog.

The Naked Computer

by Jack Rochester
and John Gantz
William Morrow & Co.
105 Madison Avenue
New York, NY 10016
(212)889-3050

The Naked Computer promises to be an eclectic and entertaining compilation of computing fact and trivia. It's got history (like why Hal sang "Daisy, Daisy"). It's got the words to the "official IBM fight song, banned since 1957." It has a serious side, too, including chapters on robotics, artificial intelligence, and computer art. Produce $15.95 and the 335-page hardcover book is yours.

The Whole Computer Catalog

Designs III Publishers
515 W. Commonwealth Avenue
Fullerton, CA 92632
(714)871-9100

The Whole Computer Catalog sounds like it wants to be all things to users. The $35, paperback, 400-plus page catalog offers information on a host of subjects. Some are: hardware, software, applications, education, careers, stores, associations, magazines, government publications, books, manufacturers, consultants, the sociological impact of computers, and "sources to find solutions to immediate problems."

Quarterly Software Reviews

Whole Earth Software Review
Box 27956
San Diego, CA 92128

From editor Stewart Brand, who brought you the Whole Earth Catalog, comes the Whole Earth Software Review, a research project/publishing venture that's to culminate later this year in yet another Whole Earth tome, you guessed it, the Whole Earth Software Catalog. Brand and staff are focusing on "the best" personal computer software, hardware, and books as objectively as possible and without, a promotional pamphlet says, any advertising whatsoever. They are winnowing the best from the rest by relying on reader opinion and experience, and an in-house research staff. The Review, its promo promises, will be enthusiastic in tone, not condescending or overly technical. It costs $18 for four issues if subscribed to; the cover price is $5 per issue.

Machine-Language Subroutines for the Color Computer

Wayne Green Books
Peterborough, NH 03458
(603)924-9471

Machine-Language Subroutines for the Color Computer, by David D. McLeod, is a reference book, not a tutorial, for computer users already familiar with assembly language programming. Its four sections are titled Preliminary Information, ROM subroutines, RAM subroutines, and Appendices. Most routines in the book can be combined with Basic programs. Entry requirements, exit conditions, and error interpretation are discussed. Included is a cassette with the source codes. It is compatible with Radio Shack's EDTASM + and Micro-Works' Macro-80C disk editor/assembler. The package sells for $29.97.

If a single goal throughout all of time can be determined for all mankind, it would be the desire to live a pleasant existence. Pleasure, happiness — more than mere survival — is what fuels our desire to live. Our history is full of wars and oppression that illustrate how far we've come and how much further we still need to go. Our attempts to attain that desired pleasant life have been wide-ranging, as evidenced by such diverse extremes as global organizations like the United Nations to localized debates over providing sufficient parking areas downtown.

The desire for a pleasant existence and our continuing search for methods to bring it about is what history is made of. Our search drives us to discovery and artistic expression. Co-existence, the first ingredient for a pleasant existence, would be impossible without an ability to communicate. The threat of all-out nuclear war seems more real today than it did a few years ago; not because our weapons are bigger and more powerful, but because the leaders of the US and USSR aren't talking together.

As society continues to elevate itself via newer, slicker, and faster technologies, our means of communication must keep pace. Until the microcomputer we depended on the telephone. Before that we used radio transmission, telegraph, fast horses, and slow boats.

Many people are asking, "Exactly what does one do with a microcomputer?" We use it to communicate with each other and with ourselves. We use it to store, manipulate and exchange information. People who dismiss the microcomputer as an adult toy are making a terrible mistake.

Reports of the multitude of privately-owned microcomputers that we hear so often are not marketing hype, nor is this a fad phenomenon that will soon die a death similar to the hoola hoop, home video systems, and Tony Orlando and Dawn. We are well into the beginning of another major epoch of human history. "The Information Revolution" will be viewed historically like the artistic Renaissance of the late 14th century and the Industrial Revolution that began in the late 1700s and continued until it collided head-on with the transistor.

In the overall scheme of things, microcomputers will probably become a subsection of a chapter on digital communication technology in a history text called *The Dawn of The Information Age*. In volume two of that yet-to-be-written historic text, microcomputer history may be compressed to a single paragraph or pair of sentences. Still, no matter how miniscule the microcomputer becomes in history's grand retrospective, its effect is quite the opposite in our daily existence. The way we conduct business, teach and learn, shop, correspond, etc., etc., has forever been changed because of the increasingly affordable microcomputer. We can watch that change happen on an almost daily basis.

Although the Information Age did not begin with microcomputers, microcomputers may prove to be the catalyst that brings the Age out of its infancy. The Information Age began with two relatively unrelated inventive feats: Daguerre's invention of photography in 1839, and Bell's telephone in 1876. Public radio broadcasting emerged in the 1920s, bringing events otherwise widely displaced by time and distance into radio listeners' present existence. In 1936 radio's sound broadcasting technology merged with the process of making pictures. The result was television, which made those far-away occurrences nearly as real and immediate to viewers as to participants. Many lesser, albeit important, advances continued to refine the art of transporting and communicating information. Concurrently, industrial technology made individual ownership of communicating devices widely affordable. Soon few houses were without a radio, and not long after a TV, and then color TV.

Technology marched on, from the tube to the transistor to the printed circuit board to the silicon micro chip, depositing radios, TVs and computers in its wake. It's difficult to believe now, but when TV broadcasting was in its infancy few people foresaw a large public acceptance of the media. It wasn't long before that assumption was proven false. Soon everyone knew the voices of Uncle Milty, and Jack's Armstrong and Benny. As the voices became faces on the television screen, listeners became viewers. Soap operas, comedians and music made the technology easy for the public to absorb.

Then this pleasant form of communication began to get serious: we watched presidents talking to reporters, racial violence came into our living rooms, and a war from a far off corner of the world continued every evening at 6:00. The electronics did not always make us feel good now. Information, for better or for worse, was communicated to America live and in color. In 1963 America buried a president, and the whole world went to his funeral.

Although Bell's telephone invention seems to have taken place many more years in the past than it actually did, modern communications technology is *that* young — merely 108 years old. In a retrospective of historical events, it is easy to realize how few years of development led up to the microcomputer. The Renaissance lasted nearly 300 years. The Industrial Revolution put in close to 200 before the transistor was born. Epochs have no time length requirements to follow, they are at the mercy of history. Without a sure-fire way to look to the future, there is no way to determine at what point we are now in The Information Age.

Realize, though, how history will transform events into matter-of-fact happenstance. I condensed modern communications history to a pair of paragraphs, the transistor to an off-the-cuff remark exemplifying the end of the Industrial Revolution, and suggested that the microcomputer would be awarded even less space in tomorrow's history books.

People commonly make two mistakes when faced with today's focus on the microcomputer: they ignore it, hoping it will fade away, or become deathly afraid they'll sleep too late one morning and miss a brand new product announcement. Now that the microcomputer has arrived and dug itself into our lives, we need to be realistic about what it means. We have reached only a juncture in human history, not a concluding episode.

— *K.L., Editor-in-Chief*

FOR...NEXT (07,84)

THE JULY ISSUE of **The Color Computer Magazine** will take a look at operating systems. We'll have a review of one of the CP/M versions out for the Color Computer, and some articles that will help you understand what an operating system is and does. Jean Plesser will present the first of two special articles for kids on their summer school break, and we'll have the last winning game from last year's Summer Programming Project. And that's just the beginning...

Explore the ancient, mystical tomb of the great Pharoah. Find the magical keys which lead you to unbelievable treasures as you out maneuver the creatures that slither and swarm about you. Super fast arcade action—this one will knock your socks off with 16 screens of incredible color and sound. $24.95 cassette, $27.95 disc. Requires 32K.

TIME FIGHTER

Pilot your MD-64 fighter through a hazardous time tunnel. Your mission is to destroy the dreaded Time Guardian who threatens the natural order of the universe. In order to reach this menace you must fight aerial dangers from strange and different time zones. $24.95 cassette, $27.95 disc. Requires 16K.

Mark Data Products

24001 Alicia Pkwy., #207, Mission Viejo, CA 92691 • (714) 768-1551

ORDERING INFORMATION: Phone your order for speedy delivery. Use your MasterCard or Visa. We also accept checks and money orders. ALL ORDERS: Please add $2.00 shipping and handling in the continental U.S. All others, add air shipping and $3,00 handling. California residents add 6% sales tax. Foreign orders please remit U.S. funds. Distributed in Canada by Kelly Software Distributors. SOFTWARE AUTHORS: Contact us for exciting marketing details.

Circle No. 73 on Reader Service Card